Assignment 1 FAQs

Generic-FAQ

1. Do I need to worry about UDP being unreliable in my code?

Ans: No need to worry about packet loss, connection instability, and so on. That means, you can consider that the message will be transferred as it is over the network.

2. How should the server choose <n_port> and <r_port>? Can I just pick a port number and hard code this into my program? Or, do I need to choose it randomly?

Ans: You can do it in any of the following two ways:

a) Start with a fixed port (e.g., 1234). If it is occupied, then try with the next one (i.e, 1235) and so on until you find a free one. Note when you are going to test whether the port is available or not (by actually trying to use it in the bind method for C++ or in the ServerSocket method for Java), it will result in exception if the port is not free. So, if-else condition checking will not work here. Rather, you need to use try-catch in such cases.

b) You can use any language specific function which will return a free port. For example, in Java, ServerSocket call with parameter 0 will give you a free port.

3. What exception/error handling do we have to do in this assignment?

Ans: You need to check the number and formats of the command line arguments passed to the server and the client.

4. What should the size of the buffer for the input string be? Will 1024 byte suffice?

Ans: 1024 bytes is sufficient.

5. Is there a limit to how big our messages can be?

Ans: Size of message will be at most 1024 bytes.

6. What is the purpose of the request code in this assignment?

Ans: Request code needs to be checked in the server. If the client does not send the exact request code, then the server should close the TCP connection to that client (in the negotiation port) and will not create the UDP port. Additionally, server can print this exception in the console.

7. What should the client do if it sends an incorrect req_code?

Ans: The client should exit gracefully (not crash).

8. How many Makefiles should be there?

Ans: If applicable (e.g., not Python), a single make file should be used to compile both client and server simultaneously.

9. Should we make server multi-threaded?

Ans: No need to do for this assignment. But this is an additional feature that you can include. Note no marks will be given for that.

10. A client might crash after creating a TCP connection with the server. Do we need to handle this exception?

Ans: No need to do for this assignment. But this is an additional feature that you can include. Note no marks will be given for that.

11. Can a string to be reversed contain the newline character or escape characters? For example ./client.sh <server_add> <n_port> <req_code> "Hello \n World"

Ans: There will be no newline character or escape character within the string.

12. Can a string to be reversed contain space?

Ans: Yes.

13. I cannot run the server.sh or client.sh scripts.

Ans: Run the following commands:

chmod +x server.sh chmod +x client.sh

14. What's the format of the <server_address> ?

Ans: Either IP address or hostnames (e.g., ubuntu2004-008) or both are fine. Include in the readme what is supported.

15. If there is only one client at a time in the system, why are we still listening on <n_port> when the client is in the stage 2?

Ans: The server should continue listening after one communication (negotiation + transaction) is done. We want to keep the connection open on <n_port> to handle future client requests. If you close/reopen again, <n_port> might not be free. By continuing to listen on <n_port>, we get rid of this issue.

16. When would the server terminate? Do we need to handle the termination of the server?

Ans: The server should stay alive unless you kill it (e.g., ctrl+c). No marks will be deducted if you don't handle the termination of the server.

17. How can I log into a particular linux.student.cs host? For example, ubuntu2004-008?

Ans: ssh userid@ubuntu2004-002.student.cs.uwaterloo.ca

18. Should "EXIT" be passed to the client as one of the input arguments when running it from the command line?

Ans: No, the command line arguments should only include the messages that we want the server to reverse. The client program should automatically send the "EXIT" message after it goes through all the input messages.

C++-FAQ

1. Is there any recommended socket library?

Ans: For C++ implementation, the socket implementation through GNU C library (using sys/socket.h header) is preferable.

2. What resource can I use if I'm trying to do the assignment in C++?

Ans:
You can go through these tutorials:

http://www.tutorialspoint.com/unix_sockets/socket_server_example.htm
http://www.linuxhowtos.org/C_C++/socket.htm
http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html

3. When I send 'elppA' to the server, I am expecting 'Apple' as the reversed string from the server. However, instead, the reversed string occasionally looks like this (getting some weird symbols added at the end of the message):

Apple?)? or Apple?O

So far, I figured that the server returns a correct reversed message and this is what's causing this:

char output[len]; // len: size of the message recvFrom(...)

// receive the msg from the server and store it to output

cout << output << endl; <-- problem on this line

This is only happening on some random messages with random sizes.

Ans: The output array is not null terminated. This causes cout to continue reading from memory until it gets to the null terminating character.

See: http://www.cplusplus.com/reference/string/string/c_str/

Java-FAQ

1. What does the Makefile do?

Ans: A Java Makefile will only compile the *.java files (e.g., server.java and client.java) and produce server.class and client.class. That is, it will run all required javac commands for the user.

Python-FAQ

1. Do we need to submit a Makefile for python?

Ans: No need to submit a Makefile with Python implementation of the assignment. Marks allocated for Makefile will be distributed among the README, coding Style, and Comments.

Ruby-FAQ

1. Do we need to submit a Makefile for Ruby?

Ans: No need to submit a Makefile with Ruby implementation of the assignment. Marks allocated for Makefile will be distributed among the README, coding Style, and Comments.