

GEORGE MASON UNIVERSITY
Computer Science Department

CS 571 – Operating Systems

Assignment 3 – Client-Server Programming using Sockets and RPC/RMI

1. **Implementing an arithmetic server using sockets** Write a program with two parts: 1) a client, and 2) an arithmetic server.

The job of the client is to accept input from the keyboard and translate strings that you type into requests to the arithmetic server. In other words, the client should translate strings such as “3 + 4” into a request to the arithmetic server. The format of the request packet sent by the client to the server is follows. Each packet consists of three fields: (i) a three-character string representing the operation to be performed, i.e. “add”, “sub”, “div”, and “mul” (ii) the first integer (iii) the second integer. Note that the size of each request packet (in bytes) will depend upon the representation of characters and integers in the language you’re using for this assignment. In C, a character is represented in one byte and an integer in four bytes.

The server should parse each packet, extract the parameters of the request, perform the requested operation and transmit the results to the client, which then prints out the results on the screen. The arithmetic server should implement the procedures *add*, *subtract*, *multiply*, and *divide*. Each of these procedures should accept two integers as input and return an integer as a result. In addition, the *divide* procedure must detect an attempted “divide by zero” and return an error code.

The format of a result packet is as follows. Each result packet consists of two fields. The first field is an integer that takes the values 0 or 1, where 1 represents a successful operation whereas 0 represents a unsuccessful operation (e.g. divide by 0). The second field is an integer corresponding to the result of the arithmetic operation or an error code representing the reason for the error. (For example, you can use the integer 1 to represent “divide by zero”.)

There are two versions of this program you have to implement

1. The client and server communicate using UDP.
2. The client and server communicate using TCP.

You have to submit to me: (a) the client program (b) the server program (c) output showing your program works correctly (you can run both the client and the server on the same computer for generating this output)

NOTES:

1. You can use Java or C/C++ for this assignment. You can reuse most of the code in the example programs available on the assignment web page.
2. It is **not acceptable** to translate integers into a string for transmitting them over the network. Your packets must follow the formats specified in the description above.

3. *Choosing a server port.* You will need to run server processes that can coexist with other people's processes in the same computer. You will need to select an agreed port number for the server to accept messages from clients. Two servers on the same computer cannot use the same local port number. You will therefore need to choose a port number that is different from that of other people.

2. **Implementing an arithmetic server using RPC/RMI** Write a program with two parts: a) a client, and 2) an arithmetic server. The client and server should communicate with one another using RPC or RMI.

The job of the client is to accept input from the keyboard and translate strings that you type into calls to the arithmetic server. In other words, the client should translate strings such as "3 + 4" into an *add()* call to the arithmetic server. The server should perform the work requested by the client and transmit the results to the client, which then prints out the results on the screen.

The arithmetic server should implement the procedures *add*, *subtract*, *multiply*, and *divide*. Each of these procedures should accept two integers as input and return an integer as a result. In addition, the *divide* procedure must detect an attempted "divide by zero" and return an error code.

You have to submit to me: a) the client program b) the server program c) the input file to *rpcgen/rmic*, and d) a script file showing all the steps involved in compiling and running your programs, i.e., compiling a file using *rpcgen* (or *rmic*), linking the generated files together, and running the programs (you can run both the client and the server on the same computer for generating this script)

NOTE If you're doing this assignment in C/C++, please read the chapters on "Introduction to Remote Procedure Call" and "rpcgen Programming Guide" in the Solaris Network Interfaces Programmer's Guide (available on the class home page) before attempting this assignment. If you're using Java RMI for the assignment, please read the online Java RMI tutorial before doing the assignment.