CSE5306, Distributed Systems
Fall 2023, Project 1
Due date: 11:59 pm Sept. 26, submission through Canvas

Please read this:
Two students form a team and turn in one submission.
Total points possible: 100 pts.
Please add the following statement at the beginning of your submission.
I have neither given nor received unauthorized assistance on this work.
Signed: Date:

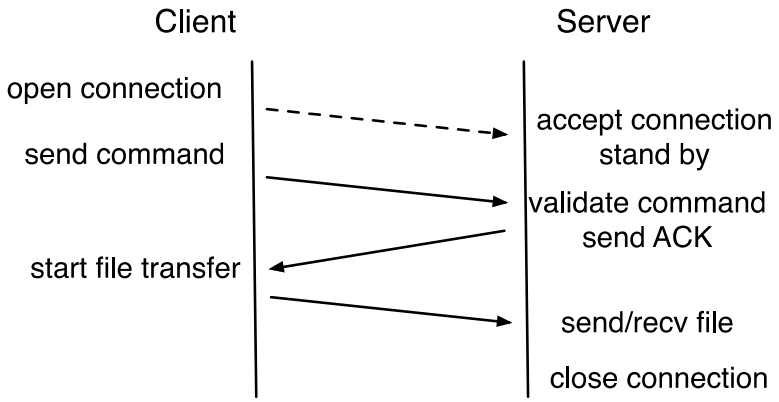
Introduction

In this programming project, you will practice programming client-server communications and remote procedure calls (RPCs). In the client-server communication assignment, you will implement a simple web server using various communication schemes, such as UDP and TCP. In the RPC-server assignment(s), you will implement three types of RPC schemes, i.e., synchronous, deferred synchronous, and asynchronous RPC, and compare their performance by varying the amount of computation on the RPC server.

You can use **any** programming language to implement the servers, though some approach is easier to implement using a specific language. The C programming language and Java are recommended.

Message-oriented client-server communication

Use the following protocol to define message-oriented communications.



To establish client-server connections using sockets, the server side creates a socket and binds it to port number 8080 (c program):

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(8080);
bind(sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
```

The server then listens to the socket and calls function accept(). The server in put to sleep until an incoming client request establishes a connection with the socket. The accept function wakes the server up and returns a socket representing the established client-server connection.

```
fd = accept(sockfd, (struct sockaddr*)NULL, NULL);
```

Assignment-0 (10pts)

Set up a virtual machine on your laptop or desktop and install a Linux operating system within the VM. You can opt for any Linux distribution, though Ubuntu is recommended. Utilize the Linux environment for testing your programs. Furthermore, familiarize yourself with the process of creating two Docker containers within the virtual machine and establishing connectivity between them via a container overlay network. In the following assignments, we will presume that the client and server run separately within two Docker containers, communicating with each other through the overlay network.

Assignment-1 (20pts)

Implement a basic single-threaded web server that delivers small-sized images upon client requests. For simplicity, you do not need to implement a full-fledged HTTP server and worry about parsing the HTTP headers. The worker needs to rename the requested image to a client-specific name before it is sent back to the client. For instance, you can use the file descriptor value as a suffix to rename a requested image. Implement the server using UDP and TCP, respectively.

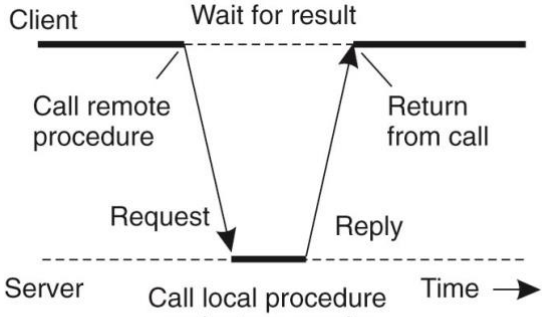
Assignment-2 (20pts)

Based on the single-threaded server, implement a multi-threaded web server. To test the multi-threaded server, the client also needs to be changed to support launching multiple concurrent requests, each from a separate worker thread. Implement two versions of the multi-threaded server, i.e., UDP and TCP, and evaluate their performance by varying the number of concurrent requests. Select a performance metric for evaluation, e.g., average request response time or throughput, and generate plots to demonstrate the differences between UDP- and TCP-based web servers.

Remote procedure call (RPC) based communication.

Assignment-3 (25pts)

Use the following design of the synchronous RPCs to implement a computation server. The server supports four RPCs: `foo()`, `add(i, j)`, and `sort(arrayA)`. You can place an arbitrary amount of computation, e.g., a for loop with a specified number of iterations, in RPC `foo()`. These RPCs represent different ways to pass the parameters to the server. You need to implement a client stub to pack parameters into a message sent to the server and a server stub to unpack the parameters. You are NOT allowed to use Java remote method invocation (RMI) or RPC in other programming languages to implement the RPC-like communication.



Assignment-4 (25pts)

Re-implement the computation server using asynchronous RPC. For asynchronous RPC, the server immediately acknowledges an RPC call before it actually performs a computation. The result of the computation is saved in a table on the server, which can be looked up by the client for the RPC result. The design of the client will be slightly different from that in the synchronous RPC. Instead of waiting for a synchronous RPC to return, the client using an asynchronous RPC switches to other computations and queries the server for the RPC after the computation is completed. Compare the asynchronous RPC server with the synchronous one using the `foo()` RPC. By changing the amount of computation in `foo()` and the amount of computation the client performs while waiting for the RPC server, show the benefit of asynchronous RPC over synchronous RPC. You need to design experiments and select a performance metric to demonstrate the benefits.

Deliverables

The deliverables include the source code of the client-side and server-side programs, a README file containing instructions on how to compile and run your program, and a report that briefly discusses the experimental results. Place all the required documents into a zipped folder and submit it via Canvas. Make sure you clearly list your names and student IDs in the report.