CMPT 431: Distributed Systems

Assignment 1, Spring 2019

This assignment is to be done in groups of 2 students!

In distributed systems, a program might need to call another function or method on a remote computer. For example, a C program on one computer might call the C function *get_stock_value(char* stock_symbol)* which exists not locally, but on a remote computer that has access to the current stock values. A Java program might need to do the same thing, although in this case it will be calling a method attached to a remote Object, since Java is object oriented. For the purposes of this assignment, we refer to these calls as Remote Programming Calls.

For this assignment, we will design a simple clock synchronization scheme between a client and a server, which you will implement with a Remote Programming Call. The client and the server are two separate machines. The client remotely calls a function on the server that returns the server's system time. The client then simply sets its own time to the system time of the server, so that both machines are synchronized. Later in the course we will see why synchronization is important for distributed systems.

Obviously there is a network delay, by the time the client receives the server's response. We must know this delay in order to set the clock with precision. Measuring one-way delay accurately is very challenging, and beyond the scope of this assignment. For this assignment, let us make the (generally incorrect) assumption that the delay from the server to the client, d_{sc}, is one half the client-to-server-to-client round trip time, *RTT*; i.e.:

$$d_{sc} = RTT/2$$

Hence, when the server responds to the client's Remote Programming Call, giving the system time at the server, t_s, the client needs to set its own time, t_c, to:

$$t_c = t_s + d_{sc}$$

Q1 (10 marks) Why is the assumption, that the delay from the server to the client is one half the RTT, not generally correct? Give two reasons.

Q2 Use Java Remote Method Invocation (RMI) or C Remote Procedure Call (RPC) to write the client and the server programs for the above scheme. There are many online RMI or RPC tutorials that you can easily find with Google. The client shall make one Remote Programming Call: to get t_s from the server. While making that call, the client shall also measure the *RTT* and consequently determine d_{sc}. Finally, the client shall use those values and set its own time to the calculated t_c.

Hint: it may be that the University blocks RPC and/or RMI ports. If so, you need to form your own wired or wireless network by directly connecting the client and the server machines together.

Show a demo to the TA, in the week of January 21, by prior appointment with the TA **(30 marks)**.
Upload your client and server codes, with inline comments explaining what the code is doing **(45 marks)**.

Q3 (15 marks) If the client does not receive a response form the server, what could be wrong? List two things that could be wrong, and explain how your program can cope with them gracefully; i.e., not just hang. There is no need to actually program your answer; just explain your approach.