Comp 734 - Assignment 2: Non-Blocking vs. RMI

Date Assigned: Tue Sep 1, 2015

Completion Date: Tue Sep 15, 20135, 11:55pm

The goal of the assignment is to show you the difference between two IPC mechanisms that are at the extremes of the IPC mechanism in performance and programmability: NIO and RMI.

Part 1: Performance of NIO Implementation

Programmatic Input: Change your implementation to allow input commands to be provided programmatically. So far, you have used the processCommand() method of the simulation command processor to programmatically change its state. As you have noticed, this method does not fire events, which is useful when you are replaying a command broadcast by another site. The method setInputString() has the same behavior except that it also fires (vetoable) events. Use this method to execute on all simulations in a session a command supplied by the program.

Local Execution: Change your implementation to allow pure local execution of commands. In this mode, your client behaves like the original simulation you extended – it does not transmit any message in response to a command.

Timings at input site: After adding the above two features (local execution and programmatic input), the next task is to use them to time the execution of a series of input commands at the site issuing these commands under various conditions. Time the execution of a series of 500 input commands under the following conditions:

1. Local execution: the simulation executes the commands locally without sending any messages to remote sites.
2. Distributed 1-User Session: the simulation sends the commands through its communication layers (that means if you have a server, the commands go to the sever, and if you have a P2P implementations, the commands go to the session layer in the client). However, there is no other simulation in the session.
3. Distributed 4-User Session: Same as above except that there are 4 simulations in the session (including the one issuing the commands).

Ideally, you would like to visually see the effects of these commands on the screen. If you are executing the move command, you can give it a small increment (e.g. 1) and provide separate commands to move in the x and y direction.

Part 2: RMI Implementation

1. Support distributed session of the kind you implemented in Assignment 1 using RMI instead of NIO.
2. Repeat the steps of Part 1 above on the RMI implementation.

Part 3: Discussion

Some of the issues you should think about and discuss for Part 2 are:

(a) What are the remote objects in the various processes involved in the RMI implementations?

(b) Are there any processes other than the simulation processes involved in mediating the connection and communication between the simulation processes? Again, you can use a server (the easy approach) or try and create a P2P system.

(c) Trace the sequence of actions that take place in each process when a user command is entered.

(d) What are some of the features you have implemented beyond what was asked?

In addition, you should report and discuss the times you measured for both NIO and RMI, explaining the differences between them if possible. In your discussion, give both the differences you would have expected and the ones you found.

Submission Instructions

(a) For part 2, create a YouTube video, demonstrating the working of the RMI-version of the program. The video can be submitted as a private or public link in piazza. If Piazza and YouTube make you uncomfortable, you can create a shared folder or use email.

(b) Answer the questions posed and submit a printout of these answers in my mail box or in class.

(c) Submit your code on Sakai (the assignment will be created before Tuesday).