Grid Computing M (GC-M)
RMI Assessed Exercise: 2005–06

Issued: Friday 20th January 2006
Submission deadline: 11am Monday 30th January

This is the second of the assessed warm-up exercises for the module, and it builds on the unassessed activity issued at the second lecture of the module, on Java RMI. The exercise will be marked as either 0 or 1. Your overall coursework mark for this module is the product (not the sum) of the marks from the two warm-up assessed exercises and the main assessed exercise, so it is important that you get a mark of 1 for this exercise. Any reasonable attempt at this exercise will gain 1 mark, non-attempts score 0, truly disastrous attempts may score 0, or trigger additional coursework to again be marked from 0 or 1 in lieu of this exercise.

Warning:

This is a solo exercise and must not be discussed with your classmates or anyone else. However, you can discuss the non-assessed code distributed previously, to ensure that you understand the ideas behind distributed programming with Java RMI. You are also welcome to inspect the answers to the DAS4 exercise 0, which are available in `/users/students4/software/public/DAS4/exercises/assessed-exercise-0`

Problem Statement:

You are to use Java and Java RMI to produce a collection of interfaces and classes which combine to produce a simple distributed system, with two distinct sorts of processes in it, which have the following properties.

The user runs their Coordinator program, which registers a remotely invocable object in the local rmiregistry and then outputs messages (by printing lines of text to its standard output) when methods on that object are called. The user also runs a number of copies of a second program, on different machines, which also instantiate a (different class of) remotely invocable object. We shall call these second programs the Elements of the distributed application. The Element objects are allowed to know, or be told via the command line, which machine is hosting the Coordinator program. The Element programs should be run on a variety of different machines, rather than all on one host; consider using ssh to connect to Linux boxes in bo709 in order to gain access to sufficient processors.

The Element objects acquire a reference to the Coordinator object and they then use that reference to give the Coordinator object a reference to themselves. Acquisition of these references is reported to the user by the Coordinator object. When ten such references have been acquired, from different hosts, the Coordinator object invokes the Element objects, passing each of them copies of the references of all of the Element objects. Thus all the Element objects now form a clique. The Coordinator object then invokes one of the Element objects at random, causing it to initiate a Spanning Tree construction.

The Element objects must create a spanning tree amongst themselves, informing the Coordinator object when this is completed. The Coordinator object will then invoke one of the Element objects at random, causing it to use the spanning tree to broadcast its identity[1] through the spanning tree, with each node reporting the identity found in the broadcast to the Coordinator object for reporting to the user.

Finally, once the Coordinator object has received and processed all of these reports, it should invoke all of the Element nodes (possibly using the spanning tree) and cause them all to terminate cleanly.

What to Submit:

Your submission should consist of hardcopy printouts of all of your .java files (use a2ps to create them), together with a single hand-written piece of paper containing: your matriculation number (but not your name), a brief status report indicating whether or not your solution works, and a short statement (max 2 paragraphs) explaining how you know what the status of your solution is. All of the sheets of paper should be stapled together, along with a "pink plagiarism sheet" (available from the Teaching Office) and submitted in the appropriately labelled locked box on the ground floor of house 16 in Lilybank Gardens. Do not use plastic covers or other binders, just staple the papers together.

[1] The identity can take any form you choose, e.g. a parameter you provide, a random number, the identity of the host machine; this detail is not important.