Lab #2: Distributed Objects

Due: April 24, 2006 11:59pm

Overview

This lab is meant to familiarize you with distributed objects. Specifically, this lab introduces the use of CORBA and Java Remote Method Invocation[1]. This introduction is brief, but should provide you with enough experience to determine if you want to use these technologies in later projects.

Overview

There are distributed systems in which the components share a set of resources that must be used in conjunction. As such, access to these resources must be synchronized as with any shared resource. One approach to such synchronization is a lock server. Processes sharing a resource can gain exclusive access to the resource by acquiring a lock on the lock server. Such a process can, in turn, release the lock to allow other processes to access the resource.

For this lab, you must implement a lock server as a distributed object. This object must support two operations. The first, `acquire`, is invoked when a process wants to acquire the lock. This method returns, once the lock has been acquired, an integer assigned by the server to identify the locking process. The second operation, `release`, is used by the process holding the lock to release the lock. This method takes as an argument an integer identifying the process. This identifier must match that of the process currently holding the lock (if it does not, then throw an exception indicating an illegal attempt to release the lock)[2].

Part 1: CORBA

Implement the lock server using CORBA. You may use any supported language, but I must be able to run your program on one of the department Unix machines (e.g., vogon or hornet). As such, if your choice of language requires an orb or IDL translator not already available on these systems, then you must provide a free one.

The server program must take the initial host and initial port for the orb as command-line arguments. It must then register the lock server object with the naming service as `LockService`.

You will need to submit all source files that you wrote as well as your IDL file.

Part 2: Java RMI

Implement the lock server using Java RMI. The server must create a registry and bind a server object to it under the name `LockService`.

The server program must take the port on which the registry will listen as a command-line argument.

You will need to submit all source files that you wrote.

Handin

Submit your source files using handin on hornet/falcon. To do this, execute

handin akeen lab2 <your files here>

Grading will be divided as follows.

Part	Percentage
1	50
2	50

[1] For this part, you will be required to use Java.

[2] Yes, this "authentication" technique is easy to spoof, but is simply meant to introduce a parameter and return type. There are actually locking techniques in which the process that releases the lock differs from the process that originally acquired the lock.