Alexandria University
Faculty of Engineering
Computer and Systems
Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: 21st Feb. 2016
Due: 28th Feb, 2016

Bulletin Board System

Objective

This assignment aims to practice working with RMI.

Description

In this assignment we will be simulating a news bulletin board system. In this system there are several processes accessing the system, either getting the news from the system or updating the news. The system will consist of a server that implements the actual news bulletin board and several remote clients that communicate with the server using the RPC/RMI; So, we will be developing a distributed client/server version of the solution using a service-oriented request/reply scheme. The remote clients will have to communicate with the server to write their news to the bulletin board and also to read the news from the bulletin board.

We will be using RMI (Remote Method Invocation) for all necessary communication instead of the Sockets (already introduced in previous courses). Also, while RMI will take care of the client threads (we don't need to handle it like sockets), we still need to provide the necessary synchronization.

Specifications

Your main class is Start.java/c/cpp; it is responsible for starting the server and the clients. First it should create a thread, which will run the server code. Then, it should start clients. This program will read a configuration file and start up the system accordingly.

The system configuration is in the file ***system.properties*** as below:

```
RW.server=192.168.1.4
RW.server.port=49053
RW.numberOfReaders= 4
RW.reader0=lab204.1.edu
RW.reader1=lab204.2.edu
```


```
RW.reader2=machine3
RW.reader3=machine4
RW.numberOfWriters=4
RW.writer0=lab204.4
RW.writer1=machine5
RW.writer2=lab204.6
RW.writer3=machine7
RW.numberOfAccesses= 3
```

RW.server is the address of the host on which the server runs. RW.server.port is the well-known port number on which the server socket will be listening. RW.reader0/RW.writer0 is the address of the host on which the reader/writer with ID 1 runs, and so on. RW.numberOfReaders is the number of readers (reader threads), RW.numberOfWriters is the number of writers (writer threads), RW.numberOfAccesses indicates how many times a reader/writer should read/write the values in shared object.

Note that the above is only a sample configuration file, you need to update it according to your machines names/IPs.

Server Specification

Our server is non-blocking that is serving several requests simultaneously without waiting any reading or writing process to be completed. The server thread needs to maintain the number of readers and writers served and when all the clients are done, should terminate gracefully.

Reader and Writer Client Specification

Clients are started as processes. Readers send their request type to the server as read and receive a packet that contains news. Writers send their request type as write and also the news to be written (its ID).

To simulate a real situation, each reading and writing operation takes a random amount of time (0 to 10000ms). A reader/writer must sleep for a random time between any two requests.

How to Start Processes on Remote Machines

Alexandria University
Faculty of Engineering
Computer and Systems
Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: 21st Feb. 2016
Due: 28th Feb, 2016

To start a process on a remote machine, you should use the remote login facility ssh in Unix systems. It accepts the host (computer) name as a parameter and commands to execute separated by semicolons.

Output Format

The server should print out its actions in the following format:

Readers:

sSeq	oVal	rID	rNum
1	-1	3	1
2	-1	0	2
3	-1	2	3
4	-1	1	4
5	-1	0	3
6	-1	0	3
7	-1	3	2
8	-1	1	2
9	-1	2	2
15	7	3	1
16	7	2	2
17	7	1	3

Writers:

sSeq	oVal	wID
10	7	7
11	5	5
12	6	6
13	4	4
14	7	7
18	5	5
19	6	6
20	4	4
21	7	7
22	5	5
23	6	6
24	4	4

Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: 21st Feb. 2016
Due: 28th Feb, 2016

Where:
sSeq: is the sequence number that reader/writer accesses the news;
oVal: is the value now saved in the object;
rID: the ID of the readers;
rNum: is the number of readers currently accessing the news;
wID: is the ID of the writer accessing the news.

A reader client should print out its actions in the following format:

Client type: Reader
Client Name: 0

rSeq	sSeq	oVal
2	2	-1
9	5	-1
10	6	-1

Where:
rSeq: the sequence number that the reader try to access the news (request sequence).
The name of the file that will be written by the clients must be **log** concatenated with **the ID of the reader** (i.e. log0, log1).

A writer client should print out its actions in the following format:

Client type: Writer
Client Name: 7

rSeq	sSeq
5	10
17	14
21	21

Where:
rSeq: is the sequence number that the writer try to access the news (request sequence).
The name of the file written by the clients must be **log** concatenated with **the ID of the writer** (i.e. log4, log5).

Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: 21st Feb. 2016
Due: 28th Feb, 2016

RMI Implementation

RMI Registry

RMI registry is a Naming Service that acts like a broker. RMI servers register their objects with the RMI registry to make them publicly available. RMI clients look up the registry to locate an object they are interested in and then obtain a reference to that object in order to use its remote methods. Of course, servers register only their remote objects, which have remote methods.

The Remote Interface

Remote methods that will be available to clients over the network should be declared. This is accomplished by defining a remote interface. This interface must extend java.rmi.Remote class and must be defined as public to make it accessible to clients. The body of this interface definition consists of only the declaration of remote methods. The declaration is nothing more than the signatures of methods namely method name, parameters and return type. Each method declaration must also declare java.rmi.RemoteException as the throws part.

System Configuration and Utility

The only addition in system.properties is that it specifies the port of rmiregistry (we assume that the rmiregistry runs on the same host as the reader/writer sever).

For example:
RW.rmiregistry.port=1099

Implementation of Remote Interface

After defining the remote interface, you need to implement this interface. Implementing the interface means writing the actual code that will form the bodies of your remote methods. For this purpose you define a class from

which you create your remote objects that will serve the clients. This class must extend RemoteObject provided by the java.rmi.server package.

There is also a subclass UnicastRemoteObject which is also provided by the same package that provide sufficient basic functionality for our purpose. When you call its constructor, necessary steps are taken for you so that you will not need to deal with them. An RMI server registers its remote objects by using bind() or rebind() method of Naming class.

RMI Clients

RMI clients are mostly ordinary Java programs. The only difference is that they use the remote interface. As you now know remote interface declares the remote methods to be used. In addition, the clients need to obtain a reference to the remote object, which includes the methods declared in remote interface. The clients obtain this reference by using lookup() method of Naming class.

Readings

- An Overview of RMI Applications: http://docs.oracle.com/javase/tutorial/rmi/overview.html

Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: 21st Feb. 2016
Due: 28th Feb, 2016

Server-Client Specifications Reminder

- The name of the program to run the system should be Start.java/c/cpp .
- Output files must be named like log0, log1, etc. These are to be plain text files.
- All the file names must be exactly as specified in this document.
- Readers should be given names 1,2,3,...,n (where n is the number of readers).
- Writers should be given names n+1,n+2,...,n+m (where n is the number of readers and m is the number of writers).
- It is sufficient for a reader client to send only the request type as read, however, a writer should send both request type write and news (its ID).
- Our news is a one shared integer where writers modify it with their ID value and readers read it.
- The server provides two numbering sequences for the clients: (1) the Request Sequence (rSeq) is the sequence number for the incoming requests and (2) the Service Sequence (sSeq) is the sequence number of the request being serviced. They are different due to the access policy that you will implement.
- The server log format is specified above.
- The client logs should be at the clients' machines and their format is specified previously.

Notes

- Develop this assignment in Java or C/C++.
- You should deliver a report explaining your design and implementation.
- We will test the assignment using multiple machines.
- The assignment is based on assignments from UFL.

Grading Policies

- No Late submission is allowed.
- You should work in groups of 2 students.
- Plagiarizing is not acceptable. Sharing code fragments between groups is prohibited and all the groups that are engaged in this action will be severely penalized. Not delivering the assignment will be much better than committing this offence.