Assignment 3
Read-Write through File Servers using RegistryServer
JAVA RMI

Deadline [Most likely it will be on 25/03/14]

The agenda of the assignment is the same as assignment 2.

- Write the whole file to the server in chunks. (Same conditions as the assignment 2, <64k reject)
- Read the whole file back.
- Using the previous APIs

But now, there will be 3 entities as follows:

1. **Registry server**: This will keep track of all instances of the FileServer
2. **File server**: We will run multiple instances of the FileServer per node. For e.g., there could be 3 servers per node. It will register itself with the RMI Registry and Registry Server.
3. **Client:** The client will first contact the registry server to know the Binding name of FileServers (The name with which you uniquely identify the object in the RMI Registry). For writes, it will send the FileWrite64K RPCs to the **master**. For reads,
   [*ADDITION***]**
   As we have to perform parallel reads, The client has to create multiple threads as many as the FileServers or as many as chunks whichever is smaller.
   Now every thread will contact one fileserver's Read64K api and make it read the chunk data eg :
   [suppose you had 2 FileServers and 5 chunks]
   FileServer 0 reads Chunk 0, Chunk 2, Chunk 4 whereas FileServer 1 reads Chunk 1 & Chunk 3.
   And write it to the same Client/output/FileName file.(As in previous Assignment)

   TIP -> Read Concept of Thread, Synchronization and RandomFileAccess.
   [*ADDITIONS OVER***]**
   This assignment will not deal with failures. So, the registry server will be started first. The FileServers will be started after that, followed by invoking the Client. Once the writes and reads are done, all servers and the registry server are killed.

Java version

RegistryServer must provide the following interfaces: boolean

RegisterServer(String name)

Each file server uses this interface to register to the registry server. Return value is as follows: true means caller has been made the master, and false means caller is the replica

Only master file server allows FileWrite64() APIs. The other file servers will return an error or throw an exception when they receive a write request.

RegistryServer also provides another interface as follows:

String[] **GetFileServers**()
Client uses this to find the names of all instances of file server.
Return value is as follows:
- null on error

An array of file servers. Each file server is identified by the name used in the call to RegisterServer(). **The first entry is the master file server**.

The RegistryServer first registers with the RMI registry, and waits for calls to RegisterServer() or to GetFileServers().

When it gets a call to RegisterServer(), it maintains the list of servers in a list.

Each **FileServer instance** has to do the following:
- **Generate a unique name**. This will be Server_<time since epoch in seconds>. While starting multiple instances of Servers, ensure that there is a gap of 2 seconds to ensure uniqueness in the name.
- **Register with RMI**
- Register with RegistryServer by calling RegisterServer() with the above name.
- Maintain a boolean that tracks if this is the master- Only **master** handles **writes**

Client program does the following:
- Looks up our registry from RMI registry
- Calls **GetFileServers**() to obtain all fileservers and the master
- Calls Naming.lookup(fileServerName) to obtain handles to each of the FileServer objects- Do writes to the master, and reads to any replica.

Upload Format

Make 3 folders

```
<Roll-no>
        |----Client
        |----Registry
        |----FileServer
```

Files

```
Client/run.sh <Name of the File> <IP Address of RMI registry> <Port No of the RMI Registry>
Registry/run.sh  <IP Address of RMI registry> <Port No of the RMI Registry>
FileServer/run.sh <IP Address of RMI registry> <Port No of the RMI Registry>
```

Keep the folders under the folder with your roll no.
<roll_no>_java_Assignment3.tar.gz
Also, we will be running MOSS. So, please do not copy.