The 2nd Programming Assignment
INF5040 Autumn 2009

The purpose of the assignment is to develop a distributed application that models a replicated bank account. The implementation should follow the ”replicated state machine” paradigm on top of group communication. The system architecture will consist of (a) the standard Spread server and (b) a client that you need to develop and link with the Spread library. The application only needs to support a single bank account with the sequentially consistent replication semantics. Each running instance of the client will represent a replica of this account. You can choose to implement the client in Java or C/C++.

The synopsis for running the client should be:
java accountReplica <server address> <account name> <number of replicas> [file name] for java, or
accountReplica <server address> <account name> <number of replicas> [file name] for C/C++.

<server address> is the address of the Spread server the client should connect to. Several Spread servers will be constantly running on the machines of our lab. The address of the Spread servers will be given to you by the teaching assistant. When deploying clients on University machines, you should use those servers instead of running your own ones. If you want to test your client outside the University, you should run your own Spread server and the clients should connect to it.

<account name> stands for the name of the account. In order to avoid name clashes between different groups, each group of students should choose a unique name when running the client. A good tactics would be to use an actual person name or id as part of the account name. All clients that a student group runs should use the same account name.

<number of replicas> is the total number of clients that will be initially deployed for <account name>. In order to simplify the assignment, the application does not need to handle dynamic addition of new replicas. On the other hand, the application should tolerate departure of individual clients (due to leaves or crashes) and continue operation when it occurs. All clients can be deployed on the same machine or different machines; it will not affect the behavior of the application. It may be a good idea to test the application with three clients as in practice, some problems may manifest themselves when the number of replicas is at least 3.

If the optional argument of [file name] is not present, the client will interactively accept commands from the user through a command line. If [file name] is present, then the client will perform batch processing of commands that it will read from [file name] and exit.

Client execution should be as follows:

1. The client should create a connection to a Spread server.
2. The client should initialize the balance on the account to 0.0.
3. The client should join a group whose name is <account name>.
4. The client should wait until it detects that all <number of replicas> clients have joined the group. To this end, it should receive and analyze messages about membership changes.
5. If the client were deployed without [file name], it should open a command line and wait for user commands. If [file name] is present, then the client will perform batch processing of commands that it will read from [file name] and exit.

The following commands should be accepted and supported by the client:

1. **balance**
   This command causes the client to print the current balance on the account.
2. **deposit <amount>**
   This command causes the balance to increase by <amount>. This increase should be performed on all the replicas in the group.
3. **addinterest <percent>**
   This command causes the balance to increase by <percent> percent of the current value. In other words, the balance should be multiplied by (1 + <percent>/100). This update should be performed on all the replicas in the group.
4. **sleep <duration>**
   This command causes the client to do nothing for <duration> seconds. It is only useful in a batch file.
5. **exit**
   This command causes the client to exit. Alternatively, the client process can be just killed by the means of the operating system.

If the client is deployed with [file name], the batch file should just contain a single command on each line.

As a reminder, the ”replicated state machine” paradigm dictates that all the replicas that do not fail go through the same sequence of changes and end up with the same balance value. The execution should satisfy sequential consistency wrt to the deposit, addinterest, and balance operations.

Development tools:

The Spread toolkit for group communication. You can use either the Java or C/C++ interface. Links for download and further details are available on the group home page.

Deliverables:

Announcements requested by the Department of Informatics:

- This assignment is mandatory and must be approved to take the INF5040 exam.
- All students must read the departmental guidelines for written assignments:
    - Norwegian: http://www.ifi.uio.no/studinf/skjemaer/erklaring.pdf
    - English: http://www.ifi.uio.no/studinf/skjemaer/declaration.doc

Deadline:
23:00 on Friday, 14th Nov. 2009