GAME OF THE ROPE*

One of the most popular traditional games is the *Game of the Rope*. Two teams of contestants face each other on a playground trying to decide which one is the strongest by pulling at opposite ends of a rope. If both teams are equally strong, the rope does not move, a standstill occurs and there is a draw; if however one of the teams is stronger than the other, the rope moves in the direction of the stronger team so much faster as the strength difference is larger and this team wins.

A variation of this game will be assumed here. A match is composed of three games and each game may take up to six trials. A game win is declared by asserting the position of a mark placed at the middle of the rope after six trials. The game may end sooner if the produced shift is greater or equal to four length units. We say in this case that the victory was won by knock out, otherwise, it will be a victory by points.

A team has five elements, but only three compete at each trial. Member selection for the trial is carried out by the team's coach. He decides who will join for next trial according to some predefined strategy. Each contestant will loose one unit of strength when he is pulling the rope and will gain one unit when he is seating at the bench. Somehow the coach perceives the physical state of each team member and may use this information to substantiate his decision.

In order to ensure rules compliance, there is a referee. She has full control of the procedure and decides when to start a new game or a trial within the game. She also decides when a game is over and declares who has won a game or the match.

Write a simulation of the activities carried out during the *Game of the Rope*. The simulation shall be based on the client-server model, with server replication, where the referee, the teams' coaches and the contestants are the *clients* and the access to the information sharing regions are the *services* which are provided to them by the *servers*.

The operations that were assigned to activities previously carried out in the information sharing regions (for the already implemented concurrent version), must now be assigned to independent requests performed on the servers, seen as remote objects, through remote method of invocation.

One aims for a solution to be written in Java, to be run in Linux under Java *RMI*, either in a concentrated manner (on a single platform), or in a distributed fashion (up to 8 different platforms) and to terminate (it must contemplate service *shutdown*). A mechanism based on *vector logical clocks* among the *clients* must be installed in order to carry out the causal ordering of operations upon the General Repository.

A *logging* file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

* Concept by Pedro Mariano

Guidelines for solution implementation

1. Specify the vector logical clock *service* which will be used.

2. Proceed to its coding in Java as a specific reference data type.

3. Specify the general organization of the servers architecture. Remember that the new APIs on access to the remote objects must include a *vector time stamp* as a parameter in each call and return a *vector time stamp* to perform clock synchronization at the client side.

4. Proceed to its coding in Java as specific reference data types.

5. Specify the general organization of the clients architecture.

6. Proceed to its coding in Java as specific reference data types.

7. Specify the mapping of the servers and the clients onto multiples nodes of the parallel machine and write the *shell scripts* which enable the deployment and the execution of the different modules the application is composed of.

8. Sketch several *interaction diagrams* which describe in a compact, but precise, way the application dynamics of your implementation.

9. Validate your solution by taking several runs and checking for each, through the detailed inspection of the *logging* file, that the output data is indeed correct.