Assignment 3

Consider the Ivy Architecture discussed in the class. In the Ivy architecture, the central manager is in charge of scheduling the read and write requests to the appropriate owner of a page. However, the central manager (CM) captures a single point of failure. In this assignment, we will implement a fault tolerant version of the Ivy architecture using the GO language.

To consider fault tolerance in the design, assume that we implement several replicas of the central managers. To maintain consistency in the data maintained by the CM replicas, a primary CM orders all the requests arriving from the clients and broadcasts this order to all secondary replicas. The primary replica in the CM is chosen via some election algorithm. Similarly, when the primary replica fails, the election algorithm is invoked to choose a new primary replica.

Using the GO language, implement the following variant of Ivy architecture:

1. **(10 marks)** First implement the Ivy architecture discussed in the class.
2. **(15+15+15 marks)** Based on the aforementioned ideas, design and implement the fault tolerant version of the Ivy architecture (the primary and secondary replicas for the CM with consistency + election to choose a primary replica + related changes in the basic Ivy protocol)
3. **(5 marks)** Argue whether your design still preserves sequential consistency (a short paragraph will suffice).

Experiments:

(15 marks) Without any faults, compare the performance of the basic version of Ivy protocol and the new fault tolerant version using requests from at least 10 clients. You should assume at least three replicas for the central manager.

(15 marks) Evaluate the new design in the presence of faults. Specifically, you can simulate two scenarios a) when the primary CM fails at a random time, and b) when the primary CM restarts after the failure. Compare the performance of these two cases with the equivalent scenarios without any CM faults.

Assignment Submission Instruction:

Please follow the submission instructions carefully. Note that I need to run your code and need to interpret the outputs generated by your code. Thus, it is important to strictly follow the submission instruction as follows:

1. Collect all submission files and compress them in a zip file. Name the submission of your homework as **PSET3_<your_student_id>.zip**
2. Include a README file in your submission that clearly explains how to compile and run your GO programs. As the homework demands different configurations in simulation, explain in your README clearly to distinguish the cases (for example, if you wish any command line features, explain them clearly).
3. If you have used any external package for GO (should not be required for this homework), kindly explain them in your README. Note that you are still required to code the protocols yourself.
4. Kindly include some information in the README on how to interpret the output of the program. Note that I will also check the source code. Thus, even though it is not required to excessively comment, a comment per GO routine

is appreciated. Basically, each GO routine can carry a comment on what it is supposed to do.

5. Any other information that you think helpful for running your code (e.g. open issues) will be appreciated too.

Note: It should be clear that the assignment does not demand the implementation of a proper consensus (e.g. Paxos). The fault tolerance implementation is a bit rough i.e. if the primary fails before it broadcasts all requests to the replicated CM, then the system may not be consistent. This is expected behavior and it is not required to fix this behavior.