

# CS 6378: Project III

Instructor: Ravi Prakash

Assigned on: April 21, 2025

Due date: May 9, 2025

**This is a project to be done in groups of two and you are expected to demonstrate its operation either to the instructor or the TA. All team members must be present during the demonstration. Along with your assignment files, please submit a file clearly stating the contribution of each team member, including who designed/implemented/tested each functionality.**

## Project Description

Let there be seven data servers,  $S_0, S_1, \dots, S_6$  and five clients,  $C_0, C_1, \dots, C_4$ . The servers are connected in a cyclic manner, with server  $S_i$  connected to servers  $S_{(i+1) \bmod 7}$  and  $S_{(i-1) \bmod 7}$ . When a client wishes to perform a read or write, it establishes a communication channel with a server chosen as per the description below. This client-server channel lasts for the duration it takes to complete the read or write operation. All communication channels are FIFO and reliable when they are operational: implemented using sockets. Occasionally, a channel may be disrupted in which case no message can be communicated across that channel. There exists a hash function,  $H$ , such that for each object,  $O_k$ ,  $H(O_k)$  yields a value in the range  $0 - 6$ .

- Under normal (fault-free) circumstances, when a client,  $C_i$  has to insert/update an object,  $O_k$ , this operation must be performed at three servers numbered:  $H(O_k)$ ,  $H(O_k)+1 \bmod 7$ , and  $H(O_k)+2 \bmod 7$ . Instead of contacting all three replicas directly, the client establishes a channel with server identified by  $H(O_k)$  and sends its requested operation to that server. Then, that server, acting on behalf of the client, ensures that the insert/update operation is performed at the three replicas. When server  $H(O_k)$  completes the insert/update operation at the appropriate servers, it sends a response to the client at which point the client can terminate the connection with the server.
- If server  $H(O_k)$  is not accessible from the client, the client then tries to access server  $H(O_k)+1 \bmod 7$ , asking it to perform the update on the two live replicas. If neither of these two replicas are accessible from the client, the client should declare an error.
- When a client,  $C_j$  has to read an object,  $O_k$ , it can read the value from any one of the three servers:  $H(O_k)$ ,  $H(O_k)+1 \bmod 7$ , or  $H(O_k)+2 \bmod 7$ . If the contacted server is down, the client tries one of the remaining servers. If all three servers that are supposed to host an object are down, the client should display an error message corresponding to the read operation.
- You should be able to simulate failure of a server,  $S_i$ , as follows:  $S_i$  terminates its communication channels with the two servers on either side of it in the cycle. When those two servers come to know of the failure (indicated by the termination of the communication channel), the one with the lower server id establishes a channel with the other with the higher server id, thus restoring the cycle. Similarly, a *failed* server can *recover* and rejoin the cycle by establishing links and sending join messages to the servers that would precede and succeed it in the cycle. Those two servers would then break the link they had between them. For the sake of simplicity, you can assume that at any given time, at most one failure or recovery takes place; and no write operation is in progress during a failure or recovery.
- As part of recovery, when a server rejoins the cycle, it communicates with at most two servers before it in the cycle, and at most two servers after it in the cycle to obtain the latest replica of all the objects it is expected to store.

In this project you are required to implement the following (and be able to demonstrate the implementation):

1. A client should be able to randomly choose any of the three replicas of an object when it wishes to read the value of the object. If a client tries to read an object that is not present (has not been inserted earlier by any client), the operation should return with an error code.
2. When a client wishes to update/insert an object into the data repository, it should be able to successfully perform the operation on at least two, and if possible all the three servers that are required to store the object.
3. If a client is unable to access at least two out of the three servers that are required to store an object, then the client does not perform updates to any replica of that object.
4. If two or more clients try to concurrently write to the same object and at least two replicas are available, the writes must be performed in the same order at all the replicas of the object, i.e., you can employ the total ordering solution you implemented in Project 2.

You will need to demonstrate that you have implemented all the requirements mentioned above.

## Point Distribution

**Design Document (20%):** Employ appropriate software engineering practices to prepare this document.

**Implementation (30%):** Source code of your well structured and well documented program. You are required to write your code in C, C++ or Java.

**Correctness (35%):** Description of the various design choices you have made, for example the hash function, the protocol for permitting writes, etc. and what are the safety and liveness conditions you set about satisfying. The description should indicate how the safety and liveness conditions are satisfied.

**Report (15%):** Results of all your experiments presented in an understandable form.

## Submission Information

The project is to be submitted via eLearning.