Lucas Azcue Abello
Alex Blay Sitges

Exercise 4: Epidemic Replication Distributed Systems

The fourth and last exercise of Distributed Systems consists of building a multi-layered distribution system with the basis of data replication strategies. To do so, we were asked to build a server, divided in 3 layers: The first layer (Core layer) had capabilities of read and write, besides having these data replication strategies: update everywhere, active, and eager replication. The second layer (Layer 1) had read only capabilities, besides having passive replication (lazy update every 10 seconds from Core Layer) and primary backup.

To simulate this system, we have started by declaring different layers, which serve only as structuring, since the functionality of each node is specified in extension of our class Server. Through this common class, we define the basic behavior of all nodes, a JSON file that stores the current value of the HashMap variable. These values will be edited through instructions from the client server to the different core nodes and propagate through the other layers. By extending from this server class, we can add the different behavior of each layer node. It is worth mentioning that the client is considered as a server as well, despite not being one of the nodes, to take advantage of the declaration of sockets.

The Core Layer will receive the transactions, group of 3 instructions, and will block new messages until all three have been processed. This block is also shared with the rest of the core layer. The write instructions (only ones that must be propagated), must follow an update everywhere, active and eager replication. Thus, when a Core Layer node receives a write instruction from the client it modifies its own variable and propagates this instruction throughout the entire layer. The other nodes of the layer will know to follow this instruction. Once a Core Layer node has received 10 instructions, both by client and other nodes, it sends the current value of the HashMap to any connected nodes of lower layers. Notice that all nodes of this layer can receive update operations (update everywhere), which ensures data consistency by blocking incoming messages (eager replication), while propagating the update operation instead of the result (active replication).

The First Layer nodes will receive the entire HashMap (passive replication). This propagation also defines the primary copy behavior, with no guarantees ensured by lazy replication. This First Layer nodes will then propagate the value of their HashMap to the next layer every 10 seconds. The Second Layer will share the same characteristics as the first layer, passive lazy replication with primary copy.

All nodes will share the current value of their HashMap though the WebSocket when an update operation is received. Thus, the core layer nodes will send this information every time a write transaction is received, the first layer every 10 of these instructions, and the second layer every 10 seconds. In addition, every update operation in any node is also written to the associated JSON file

Through the different profiles that the layers exhibit (Core vs First/Second Layer), we can observe the advantages of each one. The Core Layer adds an extra burden to the system due to the distributed synchronization process to synchronize concurrent updates. Ensures data consistency properties but limits the throughput. And data consistency is further ensured the instructions are propagated instead of the outcome. First and second layer nodes on the other hand show a single point of failure phenomena as they depend on the core layer to receive updates, which eases replica convergence. Threatens data consistency by not guaranteeing incoming operations. Transfer of the resulting state that does not ensure result of outcome. It is important to notice how these characteristics can be combined in each node, with further combinations possible that the ones applied for this exercise. Through these combinations the manager can promote certain aspects like data consistency, or propagation through the system. It is important to understand the objectives and capabilities of each system to select the properties that fit best.