# Report for the Primary Copy Replica

Home exam 2 — Distributed objects — INF5510

Johannes Akse — johanand

To avoid the problem of a client having to wait while a request is being forwarded in the system, a new thread should be spawned for every write request. Is it possible to guaranty that data Is actually updated without delivering some kind of notification to the client. Does this require the client to be coded in a specific way, that is asynchronous, with callback methods? Can this be optional?

As I understand the assignment, we are supposed to design a framework for storing different objects distributed. For me this means that the framework should be a generic one. The only functionality of the object to store is that it should be cloneable.

To account for the two mandatory testing scenarios, the name and time servers, I find it necessary that the cloneables also need to be updateable (writable) and it also should also be possible to get the current, most recently updated data (read). For the name server, this would be the lookup service explained in the assignment.

## Organization of objects

According to the assignment “The framework should be able to tell an object which other replicas exist of a given object so that each replica does not need to keep track of all others.” I read this as an indication that the framework needs some central object to maintain a list of all the replications. This might render the system more vulnerable to crashes since the distribution of responsibility isn’t totally distributed.

To be able to check which nodes are in use, i.e. has a replica, we also need a list of some sort of all the nodes with replicas. These two lists can be merged into one. There are pros and cons of merging. When merging, the framework object will become smaller. This might not be a very big issue these days due to better hardware, but moving objects will be affected by size. Since the framework object, in my solution, won’t be moved, this is not an issue. Having both the replica list and the node list as one data structure also account for more understandable code. There is only one data structure to remember the name of. On the downside of this last argument is the fact that the structure needs to be synchronized when writing to it. This will make the system slower when some threads will have to wait to write. The system also needs to do more getter and setter method calls, which will be time consuming.

Observer pattern

State pattern

Proxy pattern

Private and public methods