University of Oslo
Distributed Objects - Spring 2019
William Janoti (wcjanoti@ifi.uio.no)

Home Exam 2 Report

1 Source Files

- **typedefs.m** - type objects used throughout the application
- **generic_replica.m** - class representing a generic replica (not primary)
- **primary_replica.m** - class representing a primary replica
- **node_event_handler.m** - object to handle node events, used by the framework
- **pcrframework.m** - main program file, defines the framework and all related logic
- **name_server.m** - simple name server implementation used in testing
- **name_server_test.m** - name server test case
- **time_server.m** - time server implementation used in testing
- **time_server_test.m** - time server test case
- **Makefile** - helper Makefile to compile and run the project, the following targets are defined.
  - **compile** - compile all Emerald files (using ec)
  - **compile_planetlab** - compile all Emerald files to run in PlanetLab (using ec32)
  - **run_nameserver** - run name server test (using emx)
  - **run_nameserver_planetlab** - run name server test for Planet Lab (using emx32)
  - **run_timeserver** - run time server test (using emx)
  - **run_timeserver_planetlab** - run time server test for Planet Lab (using emx32)
  - **clean** - removes all compiled files (.x)

2 Program Structure

2.1 Main decisions

The framework (defined in *pcrframework.m*) keeps track of all replicas by using an Emerald *Directory*. The keys are the string representing the replicated object type and the values are arrays of ReplicaTypes (defined in typedefs.m), corresponding to PrimaryReplica and GenericReplica objects. That was done to make it easier to check if a given node already had a replica of a given object when moving replicas.

I decided to have a dedicated node running the framework, meaning that no replicas are hosted in the node the framework object is running. Figure 1 shows an example of a node running the framework and its replica directory and 2 nodes each hosting two replicas of difference object types.

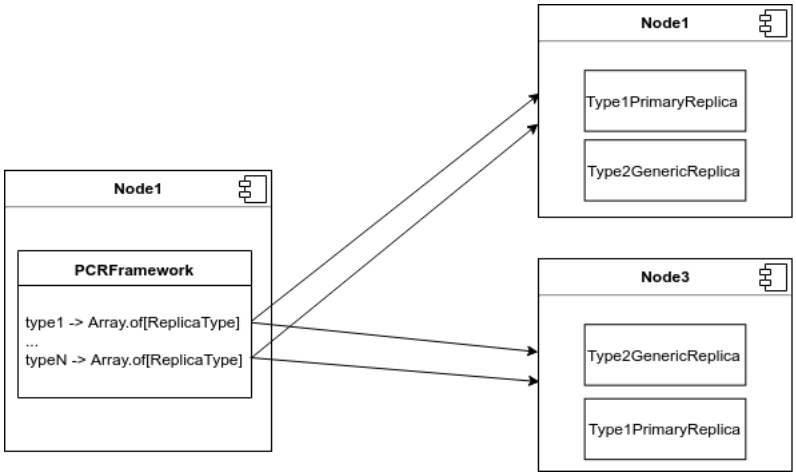

Figure 1: Program organization

All replica tracking is made by the framework. The first element in each replica array, maintained by the framework in the directory, is always a PrimaryReplica. Figure 2 tries to illustrate that.

That way, it's easy to find the primary replica based on the type of the replicated object and walk through all the replicas when needed.

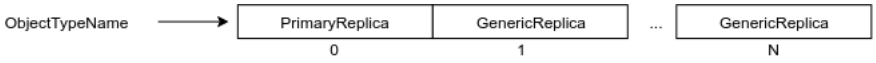

Figure 2: Replica directory array

2.2 API

The replicas act as a wrapper of the cloned object, therefore to access the object a given replica is holding a call to *read* is necessary. After manipulating an object, a call to *write* passing the updated object is necessary as well, to synchronize it with all the replicas. The listing below shows an example:

```
const testObj <- TestObject.create["Some value"]
var replicas : Array.of[ReplicaType]
% create 2 replicas of testObj
replicas <- PCRFramework.replicate[testObj, 2]
% do something with testObj
testObj.setData["Changed"]
% get any of the replicas and update it
replicas[0].write[testObj]
```

3 Testing

The testing cases was based on a basic usage of the framework, meaning creating an object, replicating it, doing some updates on it and then updating its replicas. Also, tests regarding the framework ability of redistributing replicas was done. Test outputs can be found under the **test_outputs** folder.

3.1 Name server

Used nodes: planetlab1.xeno.cl.cam.ac.uk (framework node), mars.planetlab.haw-hamburg.de and cse-white.cse.chalmers.se
Test file source code: *name_server_test.m*
Test output file: *name_server_test_output.txt*

3.2 Time server

Used nodes: planetlab1.xeno.cl.cam.ac.uk (framework node), mars.planetlab.haw-hamburg.de and cse-white.cse.chalmers.se
Test file source code: *time_server_test.m*
Test output file: *time_server_test_output.txt*

3.3 General tests

Results from tests regarding maintenance of replicas when nodes go offline can be found in the following files:

- generic_replic_recovery_test_output.txt - tested how the framework behaves when a generic replica node goes offline
- primary_replic_recovery_test_output.txt - tested how the framework behaves when a primary replica node goes offline