**Groupy: a group membership service**

**Pau Alcázar i Francesco Oncins**

**Experiments**

*i) Do some experiments to see that you can create a group, add some peers and keep their state coordinated. You can use the following code to start and stop the whole system. Note that we are using the name of the module (i.e. gms1) as the parameter Module to the start procedure. All the workers but the first one need to know a member of the group in order to join. Sleep stands for up to how many milliseconds the workers should wait until the next message is sent. ii) Split the groupy module and make the needed adaptations to enable each worker to run in different machines. Remember how names registered in remote nodes are referred and how Erlang runtime should be started to run distributed programs.*

i)

ii) Split the groupy module and make the needed adaptations to enable each worker to run in different machines. Remember how names registered in remote nodes are referred and how Erlang runtime should be started to run distributed programs.

**Handling failures**

**Failure detectors**

*Experiments. Do some experiments to see if the peers can keep their state coordinated even if nodes crash.*

**Missing messages**

*Experiments. Repeat the experiments and see if you can have the state of the workers become out of synch.*

*Open Questions. Why is this happening?*

**Reliable multicast**

*Experiments. i) Repeat the experiments to see if now the peers can keep their state coordinated even if nodes crash. ii) Try to keep a group rolling by adding more nodes as existing nodes die. Assuming all tests went well we’re ready to ship the product. There is however one thing we need to mention and that is that our implementation does not work!!! Well, it sort of works depending on what the Erlang environment guarantees and how strong our requirements are.*

i) Repeat the experiments to see if now the peers can keep their state coordinated even if nodes crash.

ii) Try to keep a group rolling by adding more nodes as existing nodes die. Assuming all tests went well we’re ready to ship the product.

**What could possibly go wrong**

*Open Questions. i) How would we have to change the implementation to handle the possibly lost messages? ii) How would this impact perfor- mance? iii) What would happen if we wrongly suspect the leader to have crashed?*

i) How would we have to change the implementation to handle the possibly lost messages?

ii) How would this impact perfor- mance? iii) What would happen if we wrongly suspect the leader to have crashed?