COP5612 – Fall 2013
Project 4 – Distributed Debugging

Alin Dobra

October 30, 2013

- **Due Date Phase I:** November 23, Midnight
- **Due Date Phase II:** Demo to Instructor on December 11-13
- One submission per group
- Submit phase I using eLearning
- **What to include:**
  - `README` file including group members, other requirements specified below
  - `project4.scala` the code for the project
  - `project4-bonus.scala` the code for the bonus part, if any

1 Problem definition

Debugging distributed programs is a real challenge. As we discussed in class, logging information about what actors send what message to what other actor can help in figuring out problems. This project consists in two phases. In phase I, you have to add logging capabilities to Scala so that information is captured about the execution. In phase II you have to use these logs to produce useful information for the user to figure out what could have gone wrong.

2 Requirements for Phase I

For phase I, you have to add logging to Scala actors. You can choose to do this any way you like but you will use the information for phase II. A typical trace will consists in simple text information in log files – one log file per actor might interfere less with the execution. This way, the logs can be processed using any tools/language.

As part of the phase I, you have to implement the logging facility and to write (or reuse) a mid-complexity actor program and to trace its execution using your facility.

Programs Write your logging code in `actorlog.scala` and write an example execution in `example.scala`. The program `example.scala` should run without arguments and produce a traced execution.

README file In the README file you have to include the following material:

- Team members
- What is working
- How you integrated logging with the actors
- What log files are produced by the `example.scala` and how to interpret them.

3 BONUS

(30%) Produce a very elegant solution in which logging is dealt with by simply inheriting from a class, without the need to litter the code with logging code.

4 Requirements for Phase II

In phase II you have to demonstrate an interesting use of the logs produced in phase I. Examples are:

1. Allow `Causeway` to read the logs you produce to be able to trace visually the execution. Add other debugging facilities such as summary statistics.
2. Visualize using `MSCGen` the message exchange. You have to deal with tens-hundreds of actors.
3. Visualize the call graph using a `.dot` file processor like Graphwiz
4. Specify conditions that need to be true for the message exchange and check that they hold (use some high level language like Prolog, ML or even Scala). A Scala based solution would be particularly nice.
5. Find other software (free or commercial) that can be used to deal with the problem

This phase can be implemented in any language you want. You just need to demo it in person (online for EDGE students, no need to sumbmit any code).