Distributed Systems Group (DS)
Department Software Technology
Faculty EEMCS
DELFT UNIVERSITY OF TECHNOLOGY

DISTRIBUTED ALGORITHMS (IN4150)
LAB EXERCISES

Exercise 3a

Implementation of Randomized Byzantine Agreement

D.H.J. Epema

December 2, 2019

1 Goal

Consensus and fault tolerance are core topics in distributed systems, and algorithms for reaching consensus are fundamental to the field of distributed computing. In this exercise a randomized solution for consensus has to be implemented that works in both synchronous and asynchronous distributed systems.

2 Assignment

Implement the algorithm for Randomized Byzantine Agreement in synchronous and asynchronous systems with a completely connected network (Algorithm 5.13 in the lecture notes). The implementation can be done in Java/RMI or Python. In designing, implementing, and testing your algorithm, take into account the following issues:

1. Your program should be truly distributed in that processes in the distributed algorithm run on multiple machines (so don't use a single JVM on a single machine that simulates all processes; it is of course allowed to have a single JVM in one machine simulate multiple processes).
2. Build into your program artificial random delays before processes perform their actions.
3. First test the correctness of your program for small numbers of processors.
4. Include different failure patterns for the faulty processes, e.g., by having them never send any message at all, or by having them flip a coin for every potential message whether to actually send it or not, and if so, for the contents of the message.

5. Also run your program with a number of faulty processes that is at least equal to one-fifth of the total number of processes, and check that then your program does not (always) give correct results.

6. Try to drive the execution of your program to a number of processes that is as large as possible.

3 Report

Write a short report (about 4 pages) in which you list for each test case that you run:

- the initial values of the correct processes
- the number of rounds used for reaching consensus
- the random values used by the correct processes in every round in which they cannot reach a decision

Also report what happens when running your program with a number of faulty processes that is too high. Don't include the algorithm itself or its explanation in the report.