Parallel and Distributed Systems Group (PDS)
Department Software and Computer Technology
Faculty EEMCS
DELFT UNIVERSITY OF TECHNOLOGY

LAB EXERCISES DISTRIBUTED ALGORITHMS (IN4150)

Exercise 3

Implementation of Byzantine Agreement

D.H.J. Epema and M.N. Yigitbasi

April 16, 2012

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. This exercise aims at achieving a thorough and detailed understanding of reaching consensus in distributed systems by means of implementing the first algorithm for this purpose ever to be created.

2 Assignment

Implement the Lamport-Pease-Shostak algorithm for Byzantine Agreement without authentication in synchronous systems with a completely connected network (Algorithm 5.6 of 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. The algorithm is designed for synchronous systems, and you will necessarily be implementing it in an asynchronous system. Implement the algorithm in such a way that all messages sent will be received in the same round with a high probability. Also record the sending of every message, and check after the completion of your program that all messages were indeed received in time.

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-third 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 in which you list the test cases of your program including the numbers of messages sent/received (and the upper and lower bound on the expected number of messages). In particular, whereas of course the final result of your program is consensus, for the cases in which the numbers of faulty processes is too high, report the input values of the correct processes and their decisions.