Distributed Algorithms Lab 3 report

Stefano Tribioli 4347269 Casper Folkers 1314130

June 16, 2014

1 The assignment

The assignment asked to implement a randomized byzantine agreement algorithm on a distributed system. This algorithm is designed to give a distributed system, where some processes may be faulty, a way of reaching consensus without taking indefinitely long. Also because of the randomized component, the algorithm can run on asynchronous systems as well as synchronous systems. To hold on to this promises the number of faulty processes has to be lower than one fifth of the total number of processes. In the next sections we will show our implementation is correct by subjecting it to different tests.

2 Test setup

To verify if our implementation is correct, we ran some tests. The test are designed to check on some boundary conditions. The variable parameters are the total number processes (n), the number of disloyal processes (f) and the type of faulty processes. table 1 shows the parameters for the different tests we ran. Each test is run for each type of faulty process to check the correct execution.

	n	f
All loyal	3	0
5f < n	8	1
5f = n	5	1
5f > n	5	2
large n	>=50	0

Table 1: Parameters for the different test

The last tests consists of multiple runs with an increasing number of processes, to estimate the number of rounds it takes to reach a decision.

3 Results

Below the different results for the tests mentioned in previous chapters are shown. To save space only one result per test is shown.

3.1 All loyal test

Table 2 shows the result of a test run with all loyal processes. Because of the small number, consensus is reached in the first round.

round	process 1	process 2	process 3
intial value	false	false	true
decision	false	false	false

Table 2: results for the test where all processes are loyal

3.2 5f < n test

Table 3 shows the result of a test run with only 1 disloyal process. The results show that the algorithm does reach consensus for this condition in 4 rounds.

round	process 1	process 2	process 3	process 4	process 5	process 6	process 7
intial value	true	false	true	false	false	true	true
1	true	true	true	true	false	false	false
2	false	true	false	false	true	true	true
3	true	true	false	true	true	true	true
decision	true	true	true	true	true	true	true

Table 3: results for the test where 1 process is disloyal, compared to 7 loyal processes

3.3 5f = n test

Table 4 shows the result of a test run where the condition 5f < n is just not met. Now the algorithm may reach a consensus in the very first rounds, like in the test shown. Otherwise, one or more processes reaches a decision, without the others being able to reach it in the following round and therefore starting to wait indefinitely.

The outcome will depend on the initial conditions: if (almost) all processes start with the same value, they will reach the right decision immediately; when the possible intial values are more equally distributed, the faulty processes can wrongly convince a subset of processes that a decision was reached.

round	process 1	process 2	process 3	process 4
intial value	true	true	false	true
1	false	false	false	true
2	no random value	no random value	no random value	no random value
decision	false	false	false	false

Table 4: results for the test where 1 process is disloyal, compared to 4 loyal processes

3.4 5f > n test

When the number of disloyal processes is increased to be a significant amount of the total number of processes the algorithm most of the time doesn't reach consensus within a reasonable amount of time.

This is probably due to the fact that when a large number of processors is faulty (f is high) the treshold for making a decision rises and the chance the process will create a new random intermediate decision rises as well. this way the chance of all processes reaching consensus gets really small.

3.5 Large n test

Table 5 shows the result for tests where n gets increasingly large, to save on data we only show the amount of rounds it took to reach consensus for the different process counts.

number of processes	Rounds untill consensus			average
50	141	74	108	108
60	147	576	1077	600
70	4136	2844	1529	2836

Table 5: results for tests with increasingly larger N

The results are too few to be an exact representation of the expected value, but they do show the exponential relation between number of processes and rounds to reach consensus.