Distributed Algorihtms Lab 3 report

Stefano Tribioli Casper Folkers

June 16, 2014

1 The assignment

The assignment asked to implement a randomized byzantine agreement algorihtm 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 synchoronous systems. To hold on to this promises the algorihtm does assume the number of faulty processes is lower than one fifth of the total number of processes.

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 paramters for the diferrent 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	100	0

Table 1: Parameters for the diferrent test

3 Results

Below the different results for the tests metioned in previous chapters are shown. to save on 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 ($5f = n$). Now the algorihm reaches a consensus, but not the expected one given the intial values.

3.4 5f > n test

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

round	process 1	process 2	process 3	process 4	process 5
intial value	true	true	false	false	true
1	false	false	false	false	true
2	no new value	no new value	no new value	no new value	no new value
decision	false	false	false	false	false

Table 4: results for the test where 1 process is disloyal, compared to 5 loyal processes

3.5 Large n test

table 5 shows the result of a test run where N = 100, to save on data we only show the amount of rounds it took to reach consensus for a couple of test runs

	Rounds untill concensus
test 1	
test 2	
test 3	
average	

Table 5: results for test where we have 100 processes