ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2014 Prof. R. Wattenhofer

# Distributed Systems Part II

## Exercise Sheet 3

**Extra Exam Points**
As mentioned in the lecture, for getting the head start points in the exam you need to submit a potential exam question and the corresponding solution. Your question should fulfill the following requirements:

- It should be non-trivial (i.e., not a pure knowledge question).
- It should be posed in a precise way.
- The solution should be correct and understandable.

Please send your question for the first chapters until December 5, 2014 to:

*distsys@tik.ee.ethz.ch.*

We will check your submissions and either reply with ACK or NACK until December 12. In case you get an ACK you will receive the extra points during the exam.
If you get a NACK, don't despair, you have a second chance to receive the extra points by sending us a new submission, until December 19, 2014.

## 1 Consensus in a Grid

In the lecture you learned how to reach consensus in a fully connected network where every process can communicate with every other process. Now consider a network that is organized as a 2 dimensional grid such that every process has up to 4 neighbors. The width of the grid is $w$, the height is $h$. The grid is big, meaning that $w + h$ is much smaller than $w \cdot h$. While there are faulty and correct processes in the network, it is assumed that two correct processes are always connected through at least one path of correct processes. In every round processes may send a message to each of its neighbors, the size of the message is not limited.

**a)** Assume there is no faulty process. Write a protocol to reach consensus. Optimize your protocol according to speed.

**b)** How many rounds does your protocol require?

**c)** Assume there are $w + h$ faulty processes. The faulty processes may die any time, but may not send wrong messages. In a worst case scenario, how many rounds does the protocol require now?

**d)** Assume there are $w/2$ Byzantine failures. How could they sabotage your protocol? Only a general idea is required, do not go into details.

## 2 Consensus in a Hypercube

Some networks are organized as a hypercube. There are $n = 2^m$ processes and each process can communicate with $m$ other processes.

**a)** Modify the King algorithm so that it works in a hypercube. Optimize the algorithm according to resilience.

**b)** How many failures can your algorithm handle? (Assume Byzantine processes can neither forge nor alter source or destination of a message.)

**c)** How many rounds does this algorithm require?

## 3 Consensus with Byzantine Failures

Does a 2-resilient algorithm without authentication for 6 processes exist? Write it down or sketch a proof for its non-existence.

## 4 Consensus with Authentication

In the lecture an algorithm using authentication to reach consensus in an environment with Byzantine processes was presented. See chapter 1, slide 132 ff for more details.

**a)** Modify this algorithm in such a way that in handles arbitrary input. Write your algorithm as pseudo-code. The processes may also agree on a special "sender faulty"-value.

Hint: implement `value` as a set, work with the size of the set.

**b)** Prove the correctness of your algorithm.

## 5 Asynchronous Consensus with Randomization

In the lecture a randomized algorithm reaching consensus in an asynchronous system with Byzantine failures was presented. See chapter 1, slides 137 ff for more details. Assume that only crash failures but no Byzantine failures can occur. A crash can happen anytime and broadcasts may not be completed. Crashed processes do not recover.

**a)** How many crash-failed processes can this algorithm handle?

Hint: Have a close look at the proofs for the validity condition, agreement, and termination.

**b)** Modify this algorithm to handle more crash failures.

**c)** How many crash failed processes can your modified algorithm handle?