ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2010 B.Keller, Prof. R. Wattenhofer

Distributed Systems Part II

Exercise Sheet 2

1 Consensus with the Aid of a Wall

Alice and Bob live in the same town. Once a year they want to meet but they don't want to be seen together in public. So they want to meet at a secret place which one of them chooses. They know a wall in town which is painted white. In addition they know a painter who paints the wall in the color they wish and sends the person who gave him the order a "before and after" picture of the wall. (Of course they color-coded each possible meeting place in advance.)

a) Design an algorithm which ensures that Alice and Bob meet at the same place

b) Can you expand your algorithm in such a way that it still works if Charlie wants to meet them as well?

c) How many persons can meet each other, if the wall is in front of the painters' shop and why?

2 Consensus through "Fetch and Multiply"

A friend of yours is convinced to have found a great algorithm to find consensus for 13 processes. His algorithm relies on a method called "Fetch and Multiply" which is described below. What would you tell him, if he asked you for your opinion?

```
public class RMW {
    private int value;

    public synchronized int FAM(int x) {
        int prior = this.value;
        this.value = this.value*x;
        return prior;
    }
}
```

3 Consensus with Byzantine Failures

Does a 2-resilient algorithm for 6 processes exist? Write it down or sketch a proof for its non-existence.

4 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.

5 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?