Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Prof. Dr. Peter M. Fischer
Y. Maus
May 20, 2015

Distributed Systems, Summer Term 2015
Problem Set 3 - Solution

The assignment is due on Tuesday, June 09, 2015, 14.15h. You can either hand it in electronically (*yannic.maus@cs.uni-freiburg.de*) or hand it in at the tutorial session itself.

Exercise 1: Clock Synchronisation

Consider the following variation of the clock synchronisation algorithm $\mathcal{A}^{max}$ from the lecture. In the lecture (and in the last exercise), on the event of receiving a neighbors time, a node updates its local time to the maximum of its local time and the received time. The variation of the algorithm cannot directly update its time to the maximum, but it stores the maximal time it has received so far and increases the speed of its clock by a factor of two as long as its actual clock value is less than the stored maximum.
The maximal skew of $\mathcal{A}^{max}$ is $Dd(1+\rho) + 2\rho T$, where D is the diameter of the network, d is the maximal message delay, T is the interval in which each clock value is at least forwarded once and all hardware clocks run at speeds in the interval $[1-\rho, 1+\rho]$ (note that we assumed $d = 1$ in the lecture).
Show that the skew of the considered variation of $\mathcal{A}^{max}$ is at most $(Dd + T)(1 + \rho)$.
Hint: A proof is similar to the proof for the skew of the clock synchronisation given in the lecture.

Solution

Let $M(t)$ be the value of the maximal clock in the system. Note that $M(t)$ is not always realized by the same node. $L_u(t)$ is the value of the local clock at time t and $M_u(t)$ denotes the maximal value that node u has heard of at time t.
The processor with the largest clock value cannot have heard of a larger clock value (because it does not exist) and thus it runs at single speed. Hence we obtain the following two claims.
Claim 1: $\frac{d}{dt}M(t) \leq 1 + \rho$.
If you have the maximum clock value you cannot have heard about a larger clock value, that is, your rate is at most $1 + \rho$.
Claim 2: $M(t - \Delta) \geq M(t) - \Delta(1 + \rho)$.
This is a direct conclusion from claim 1.
Claim 3: For all v: If $M(t) - L_v(t) > (Dd + T)(1 + \rho)$, then node v runs with rate 2.
Let $M(t) - L_v(t) > (Dd + T)(1 + \rho)$, then

$$M(t) > L_v(t) + (Dd + T)(1 + \rho) \tag{1}$$

With a similar reasoning as in the lecture

$$\begin{aligned} M_u(t) &\geq M(t - Dd - T) \overset{\text{(Claim 2)}}{\geq} M(t) - (Dd + T)(1 + \rho) && (2)\\ &\overset{(1)}{>} L_v(t) + (Dd + T)(1 + \rho) - (Dd + T)(1 + \rho) && (3)\\ &= L_v(t). && (4) \end{aligned}$$

Claim 3 shows that any clock which is further behind the maximal clock value than $(Dd + T)(1 + \rho)$ runs with rate 2.
Claim 3 combined with Claim 1 shows that if the distance between the lowest and the highest clock values is greater than $(Dd+t)(1+\rho)$ the clock with the lower value has a faster rate than the clock with the larger value, i.e., the distance between them decreases. Thus the skew is at most $(Dd+T)(1+\rho)$.

Exercise 2: Weak Agreement

You are given the following problem for two processes to find a weak agreement.

- Input/Output: Both processors have inputs and compute outputs from the set $\{0, 1\}$.
- Agreement: Possible outputs are $\{0, 1, *\}$, anything except the combination $\{0, 1\}$ is allowed.
- Validity: If both inputs are the same, they need to output this value in the case of no error.
- Termination: Non-failing processes need to terminate after a finite number of steps.

Design a wait-free algorithm/protocol that solves the given problem using only atomic read/write registers and prove the correctness of your algorithm (*hint:* there is a solution which uses only two registers).

Solution

We have two processes 1 and 2 and two variables A and B, which are initialized with value -1. The input value of process 1 is x, and the input value of process 2 is y.
Code for process 1:

```
write(A, x)
if (read(B) == 1-x) then
    decide *
else
    decide x
```

code for process 2:

```
write(B, y)
if (read(A) == 1-y) then
    decide *
else
    decide y
```

A process either decides to use its own input or $*$. Note that $read(A) == 1 - y$ only evaluates to true if process 1 has successfully written x and $x \neq y$.
To guarantee validity and agreement we need to show that the following holds in the error free case:

- The combination $\{0, 1\}$ is not possible.

  Let us take a closer look at the two write operations of process 1 and 2. One of them is first, w.l.o.g. let the one of process 1 be first. Then the read operation of process 2 is after the write operation of process 1 and process 2 can see the input of process 1. If both inputs are not the same, process 2 will decide $*$.

- If both have the same input they have the same output

  This is true indeed, as one can only decide $*$ if you know that the other process has the a different input from yourself.

Exercise 3: Consensus I

Alice and Bob live in the same town. Once a year they want to meet but they do not 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 with a single color in advance.)

1. Design an algorithm which ensures that Alice and Bob meet at the same place.
2. Can you expand your algorithm in such a way that it still works if Charlie wants to meet them as well?
3. How many persons can meet each other, if the wall is in front of the painters' shop and why? (You can assume that the painter immediately starts painting after receiving an order)

Solution

1. The algorithm looks as follows.

```
Choose the color of the place you want to meet.
Go to the Painter and instruct him to paint the wall in the corresponding color
Look at the 'before and after picture' which you get from the painter.
if the wall was white before
   the meeting place is the one you have chosen
else
   the meeting place is the place according to the 'before color'
```

2. No, what Alice and Bob can do is known as the RMW primitive-swap, which has consensus number two.
3. If the wall is in front of the painters' shop it is basically the same as the RMW-primitive Compare and Swap (because you would see if the painter is already painting, or if someone is in the shop and instructing the painter to paint) which has consensus number ∞. This means, that infinitely many persons could meet. The Algorithm would look like this:

```
Choose the color of the place you want to meet.
Go to the Painter
Look at the wall in front of the painters' shop
if the wall is white
    enter the shop and instruct the painter to paint the wall in your color
    the meeting place is the one you have chosen
else
    the meeting place is the place according to the color of the wall
```

Exercise 4: Consensus II

A friend of yours is convinced to have found a great algorithm to solve 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;
    }
}
```

Solution

The presented procedure is commutative and thus the consensus number has to be smaller than or equal to two. So I can easily convince my friend that his algorithm does not solve consensus for 13 processes (by using a result from the lecture).