ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

FS 2009 Prof. R. Wattenhofer / C. Lenzen / T. Locher

Principles of Distributed Computing
Exercise 5

1 Shared Sum

In the lecture, we discussed how shared registers can be employed efficiently to allow each process to announce a value to all other processes. Now we look at a different scenario: Each process p_i computes a local variable x_i and we want to make the sum $x := \sum_{i=1}^{n} x_i$ available to all processes.

We want to guarantee the following: If a process updates x_i, it should first ensure that x is updated accordingly before proceeding.

a) Give a solution using a single shared register supporting the fetch-and-add operation with a constant update and access complexity. If possible, prevent both lockouts and deadlocks.

b) Give a solution using a single compare-and-swap register, also with constant access complexity. If successful, an update should need a constant number of steps (otherwise the process may retry). Are lockouts excluded?

c) Give a solution using a single load-link/store-conditional register. Compare it to the preceding solutions.

d) Assume now that the return value of compare-and-swap is not whether the operation succeeded, but the value stored in the register after the operation. Can the problem still be solved? Proof your claim!

2 Space Efficient Binary Tree Algorithm

Algorithm 24 from the lecture requires to store a complete binary tree of depth $n-1$, resulting in exponential memory requirements.

Suppose Algorithm 24 is modified the following way: Whenever a process leaves a splitter with result **left** or **right** it flips a coin to replace this result by **left** or **right** with probability 1/2 each.

a) Bound the expected number of hops of a process until it leaves a splitter with **stop**, depending on the number k of active processes starting at the root of the tree.

Hint: Try the same approach as used to bound the expected running time of Algorithm 18, but on the number of processes decending a specific path in the binary tree.

b) Infer a bound on this number that holds w.h.p. (with high probability, c.f. script).

Hint: Use Chernoff's bound!

c) Conclude that the depth of the subtree induced by the marked nodes is w.h.p. in $O(\log k)$. How much memory has to be allocated to exclude a segmentation fault w.h.p.?

Hint: Make clever use of the definition of w.h.p.!