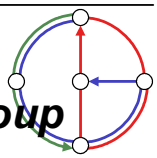
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group



HS 2009 Prof. Dr. Roger Wattenhofer, Thomas Locher, Remo Meier, Benjamin Sigg

Distributed Systems
Theory exercise 6

Assigned: December 11, 2009
Discussion: none

1 ALock2

Have a look at the source code below. It is a modified version of the ALock (slides 8/42 ff).

```
public class ALock2 implements Lock {
  int[] flags = {true,true,false,...,false};
  AtomicInteger next = new AtomicInteger(0);
  ThreadLocal<Integer> mySlot;

  public void lock() {
    mySlot = next.getAndIncrement();
    while (!flags[mySlot % n]) {}
    flags[mySlot % n] = false;
  }

  public void unlock() {
    flags[(mySlot+2) % n] = true;
  }
}
```

a) What was the intention of the author of "ALock2"?

b) Will ALock2 work properly? Why (not)?

c) Give an idea how to repair ALock2.

Hint: don't bother about performance.

2 MCS Queue Lock

See slides 8/56 ff.

a) A developer suggests to add an `abort` flag to each node: if a process no longer wants to wait it sets this `abort` flag to `true`. If a process unlocks the lock, it may see the `abort` flag of the next node, jump over the aborted node, and check the successor's successor node. Modify the basic algorithm to support aborts.

   Optional: sketch a proof for your answer.

   Hint: Be aware of race-conditions!

b) Assuming many processes may abort concurrently, does your answer from a) still work? Explain why. If it does not work: modify your algorithm to allow concurrent aborts.

   Optional: sketch a proof for your answer.

c) Instead of a `locked` and an `aborted` flag one could use an integer, and modify the integer with the CAS operation. What do you think about this idea? How is the algorithm affected? How is performance affected?

d) The CLH lock (slide 8/49) is basically the same as an MCS lock. Conceptually the only difference is, that a process spins on the `locked` field of the predecessor node, not on its own node. What could be an advantage of CLH over MCS and what could be a disadvantage?

3 Linked-Lists

a) Write a proof for: a linked-list using fine-grained locking (as described on slide 8/76 ff) does not deadlock.

b) Lazy synchronization: can the `contains` method return a false answer when searching for x because processes concurrently try to add and remove x? Make a reasonable assumption how `add` works.

c) Optimistic synchronization: describe a scenario where a process is forever attempting to delete a node.

d) CAS: think about how to implement the `add` method. Write the interesting parts as pseudo-code.