

## Excercise Sheet 9

(To be processed until 08.01.)

## Lecture Distributed Systems Winter Term 2025/26

### Exercise 1: Ring Algorithm

In the lecture (chapter 7.1) a ring algorithm was presented as one sort of election algorithm. The animation in the script shows two ELECTION messages circling simultaneously. It is not a problem if there are two such messages, but it would be more elegant if one could be removed.

Develop an algorithm that does just that, without disturbing the operation of the basic election algorithm.

### Exercise 2: Centralized Mutual Exclusion Algorithm

- a) In the centralized mutual exclusion approach (see Lecture Section 7.2), the coordinator, upon receipt of a notice from a process that releases its exclusive access to the critical section, usually grants permission to the first process in the queue. Name another possible algorithm for the coordinator.
- b) Suppose the coordinator crashes. Is the entire system always incapable of working? If not, under what circumstances does this happen? Is there a way to avoid the problem and enable the system to compensate for crashes of the coordinator?

### Exercise 3: Algorithm of Ricart and Agrawala

The algorithm of Ricart and Agrawala has the problem that if a process has crashed and does not respond to a request from another process to enter a critical section, the missing response is interpreted as a denial of permission. One way to easily identify crashed processes is to immediately (if necessary negatively) respond to all requests. Are there situations where even this method is not sufficient?

### Exercise 4: Deadlocks - Ricart and Agrawala

A distributed system can have several independent critical sections. Suppose process 0 wants to enter the critical section A, and process 1 wants to enter the critical section B. Can the Ricart and Agrawala algorithm lead to deadlocks? Explain your answer.

### **Compulsory Exercise 5: Causally Ordered Multicast** **Submit via moodle!**

Suppose a system provides reliable multicast with FIFO ordering. One could get the idea to implement a causally ordered multicast based on this as follows:

- Each message is provided with the Lamport time stamp of the send event.

- The recipient nodes deliver incoming messages to the recipient processes in an order that is consistent with the order of the timestamps.
- a) Justify why this multicast is causally ordered.
  - b) Where is the problem with this implementation? (**Tip:** when can a message be delivered to the recipient process?) How could it be solved?
  - c) Justify why this multicast is not totally ordered. How could a total order be achieved?

## **Compulsory Exercise 6: Programming: Remote Write Protocol**

**Submit via moodle!**

In this exercise, you will use the distributed systems simulator to implement the remote write protocol for sequential consistency (see Section 9.3.2 of the lecture slides).

To do this, complete the task „[Primary Based Remote Write Protocol for Sequential Consistency](#)<sup>1</sup>“ in the wiki (accessible only from within the Uni-VPN!).

---

<sup>1</sup><https://git.bs.informatik.uni-siegen.de/dsbox/exercises/wiki/6-remote-write>