

## Excercise Sheet 8

(To be processed until 18.12.)

## Lecture Distributed Systems Winter Term 2025/26

### Exercise 1: Clock Synchronization

- a) A client tries to perform a synchronization with a time server. It has executed three requests to the server and recorded in the following table the local time of sending the request, the time returned by the server and the measured round trip time.

Which of these times should it use to set its clock? What time should it set it to? Estimate the accuracy if it is known that the time between sending a message and receiving the response in that system is at least  $8ms$ .

Sending time [hh:mm:ss]	Round-trip time [ms]	Server time [hh:mm:ss]
11:51:20.043	22	11:51:20.564
11:51:21.821	25	11:51:22.340
11:51:24.715	20	11:51:25.232

- b) If the minimum transfer time of  $8ms$  is given and the clock of a file server must be synchronized to an accuracy of  $\pm 1ms$ , what is the maximum allowed round trip time?

### Exercise 2: Happened-Before Relation, Lamport and Vector Time

Three processes A, B and C each generate the events  $a_1, a_2, \dots, b_1, b_2, \dots$  and  $c_1, c_2, \dots$

- a) Suppose the events  $a_1, a_2, a_3, a_4, b_1, b_2, c_1, c_2$  have occurred and have the following Lamport times

Event	$a_1$	$a_2$	$a_3$	$a_4$	$b_1$	$b_2$	$c_1$	$c_2$
Lamport time	1	2	3	4	1	2	1	5

What can be said about the *happened-before* relation between

- $a_3$  and  $c_2$ ,
- $a_1$  and  $b_1$ ,
- $b_2$  and  $c_1$ ?

Which statements can you make about the order of the two events in real time?

- b) The following vector times now apply to the events from a):

Event	$a_1$	$a_2$	$a_3$	$a_4$	$b_1$	$b_2$	$c_1$	$c_2$
Vector time	(1,0,0)	(2,1,0)	(3,1,0)	(4,1,0)	(0,1,0)	(0,2,1)	(0,0,1)	(4,1,2)

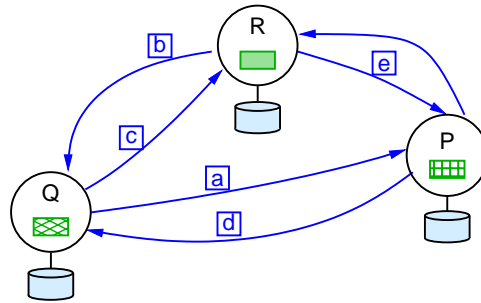
Which statements can now be made about the *happened-before* relation of the events from a)? Which statements are possible about the order in real time?

- c) How do Lamport time and vector time differ with regard to their relationship with Lamport's *happened before* relation?

**Compulsory Exercise 3: Snapshot Algorithm According to Chandy/Lamport**  
**Submit via moodle!**

In the lecture (chapter 6.4) you learned how to determine a consistent state in a distributed system according to Chandy and Lamport.

Given is a distributed system with three processes  $P$ ,  $Q$ , and  $R$ , which are connected via two unidirectional channels. The system is in the initial state shown in the figure. The messages specified on the communication connections are in transmission, that is, have been sent but not yet received.



- a) Explain the operation of the algorithm in this case and specify the global state returned by it (note: this state is not uniquely determined!). Assume that process  $Q$  initiates the snapshot algorithm.
- b) Draw a sequence diagram of the processes and indicate the cuts for the following global states:
  - the displayed initial state,
  - the state determined by the snapshot algorithm.

**Exercise 4: Snapshot Algorithm According to Chandy/Lamport**

Two processes  $P$  and  $Q$  are connected using two channels in a ring, and they constantly send a  $m$  message back and forth between them. There is only one copy of  $m$  in the system at any one time. The state of each process consists of the number of times it has received  $m$ .  $P$  starts sending  $m$ . At a certain time,  $P$  has the message and its state is 101. Immediately after sending  $m$ ,  $P$  initiates the snapshot algorithm.

In this case, execute the algorithm and specify the global state it returns.

**Exercise 5: Bully Algorithm**

The bully algorithm was introduced in the lecture (see chapter 7.1). Discuss the following questions:

- a) Suppose two processes simultaneously detect that the coordinator has failed and both perform an election using the bully algorithm. What happens?
- b) In the bully algorithm, a restored process starts an election and becomes the new coordinator if it has a higher ID than the current owner. Is this a necessary feature of the algorithm?
- c) Suggest how the bully algorithm can be customized to cope with a temporary interruption of the network (slow communication) or equally with slow processes.

**Compulsory Exercise 6: Programming: Ricart/Agrawala/Lamport Algorithm**  
**Submit via moodle!**

In this exercise, you will use the distributed systems simulator to implement the Ricart/Agrawala algorithm (see Section 8.2 of the lecture slides).

To do this, complete the task „[Ricart Agrawala algorithm](#)<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/5-ricart-agrawala>