UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Excercise Sheet 8

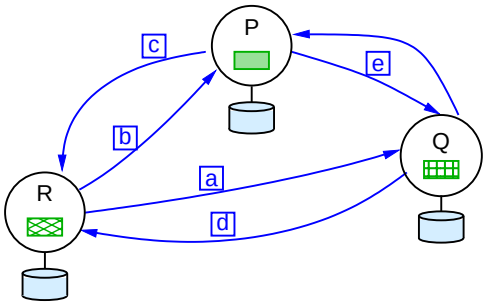
(To be processed until 12.12.)

Lecture Distributed Systems Winter Term 2024/25

Exercise 1: Snapshot Algorithm According to Chandy/Lamport (Mandatory exercise for 6 CP, 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 diagrap of the processes and indicate the cuts for the following global states:

- the displayed initial state,
- the state determined by the snapshot algorithm.

Exercise 2: 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 3: Programming: Snapshot Algorithm According to Chandy/Lamport (Mandatory exercise for 6 CP, submit via moodle!)

In the archive `u08eFiles.zip`[1] on the lecture's web page you will find the realization of a process system with several server processes. The servers receive tasks via messages, which they can process locally and pass on to other servers. There is also a client that sends tasks to one of the servers.

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u08eFiles.zip`

Extend the server code with an implementation of the Chandy/Lamport snapshot algorithm so that the total number of tasks in the system can be determined at any time. As discussed in the lecture, the difficulty is that some tasks can be "on the way" between two servers and have to be counted. To make it easier for you to transfer the algorithm, the program is based on a classical message exchange via channels, which is reproduced here using an RMI method `enqueue()` that adds a message to the recipient's queue. In order to easily verify the result of the snapshot, the jobs in the system are never completed (only passed from one server to another), so the total number of jobs does not change.

After a snapshot is finished, each server should output the status it has determined (i.e. the partial number of tasks it has determined). You do not have to program the collection of the local states (i.e. adding up the total number of tasks). Modify only the file `Server.java`.

To initiate a snapshot, a `SnapshotClient` is provided that sends a marker message to one of the servers. Test your solution with different numbers of jobs and servers.

Exercise 4: 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.

Exercise 5: 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 6: 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?