UNIVERSITÄT SIEGEN

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

Excercise Sheet 9

(To be processed until 26.06.2020)

Lecture Distributed Systems Summer Term 2020

Exercise 1: 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 2: 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 3: 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.

Exercise 4: Programming: Ricart/Agrawala

In the archive `u09eFiles.zip`[1] on the lecture's web page you will find the implementation of a process system with several server processes. At the very start, the servers receive a message from the client containing exactly one task, which they process locally. For processing, the servers must go into a critical section in which a file (`ressource.txt`) is written.

Based on the given code, implement the Ricart and Agrawala mutual exclusion algorithm so that only one server can write to the file at a time. To do this, complete the methods `lockResource()`, `unlockResource()` and `handleMutexMessage()` in `Server.java` and add any necessary declarations and initializations. To simplify things, you can assume that a server receives only exactly one request from the client, i.e., no new request can be received while the current request is being processed.

The realization of Lamport time stamps necessary for the algorithm is already completely predefined. The sender of a message can also be determined (method `getSender()` of the class `Message`).

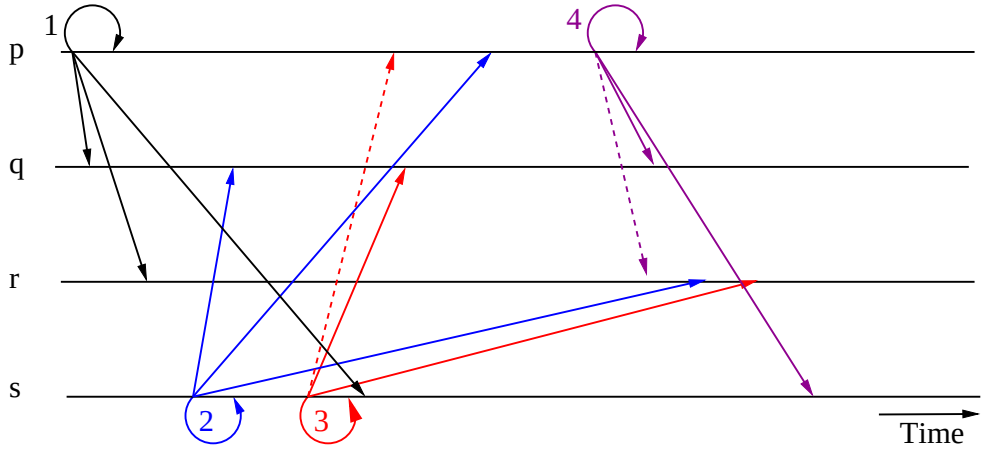
Test your solution with a varying number of simultaneous requests and a varying number of servers. The number of jobs can be specified at the client as a command line argument (maximum as many as there are servers).

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u09eFiles.zip`

Exercise 5: Multicast Order Guarantees

In the lecture different variants for order guarantees in multicast were presented (see chapter 7.3).

a) The figure shows 4 processes (p, q, r, s) and four multicast messages.



Message 4 in the example is causally dependent on message 3.
When must the delivery of the messages 3 and 4 take place, so that a FIFO order and a causal order is guaranteed for all messages? Note: You only need to draw the dashed arrows correctly. Is then also a total oder guaranteed?

b) To illustrate the multicast sorting, let's look at an "electronic blackboard" application, where users leave messages on a blackboard. Each user executes an application process for the black board. There are several discussion topics, each with its own process group. A user's process is a member of the group for the topic he is interested in, so the user only receives messages on that topic. When a user posts a message on a black board, the application multicasts it to the appropriate group.

A user X could then receive the following display from the program for the black board:

```
Bulletin board on the topic: Communication in distributed systems

Message        from            Title of message
_____________________________________________________________
   27          A. Bauer        UDP & TCP Communication
   28          C. Fritz        RPC Principle
   29          G. Meier        Re: UDP & TCP Communication
   30          K. Wim          RPC 1
   31          A. Bauer        Re: RPC Principle
   32          K. Wim          RPC 2
```

Note that the messages whose topics begin with *Re:* appear after the messages to which they refer.

Which sorted multicast methods are used in the application and where exactly? If the multicast were totally ordered, what can you say about the numbering in the left column?

Exercise 6: Causally Ordered Multicast

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?