

Excercise Sheet 10

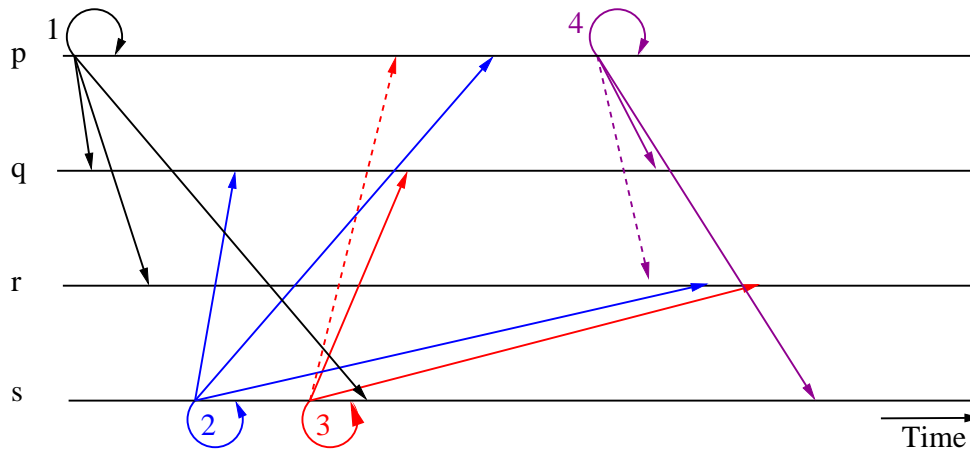
(To be processed until 15.01.)

Lecture Distributed Systems Winter Term 2025/26

Exercise 1: 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 order 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 2: Totally Ordered Multicast

To implement totally ordered multicast with the help of a sequencer, one approach is to first pass the message to the sequencer, which assigns it a unique number and then passes it on via multicast. Name an alternative approach and compare the two solutions.

Exercise 3: Totally Ordered Multicast / Lamport time stamps

In a bank implementation scenario, a totally ordered multicast is required, i.e., a multicast operation in which all messages are delivered to all recipients in the same order. Lamport time stamps are used to implement the multicast in a distributed manner. When a process receives a message, it is placed in a local queue, which is organized according to time stamps.

The recipient sends a multicast acknowledgement to the other processes. If we follow Lamport's algorithm to adjust the local clocks, the time stamp of the received message is smaller than the time stamps of the acknowledgement. After a short time, it turns out that the local queues in which the messages and notifications are located are the same for all processes, since it can be assumed that the delivery takes place in the same order as the one in which they were sent. A process can only deliver a message if it is at the beginning of the queue and has been acknowledged by all other processes.

Is it strictly required here that each message is acknowledged in order to achieve totally ordered multicasting with Lamport timestamps?

Compulsory Exercise 4: Consistency Models

Submit via moodle!

- In the lecture (chapter 8.2) a number of consistency models were presented. Why are these different forms of consistency introduced? Explain in your own words what is the most important reason to consider weakened consistency models.
- Given are the two subsequent computations in a distributed system. Justify in each case whether the storage behavior shown satisfies sequential consistency and causal consistency.

P1:	W(x)a	W(x)b			
P2:	R(x)a	W(x)c	R(y)a		
P3:	R(x)a	W(y)a	R(x)c	R(x)b	

P1:	W(x)a	W(x)b			
P2:			R(y)a	R(x)a	
P3:		R(x)b	W(y)a		

Notations:

- $W(x)a$: the value a is written into the variable x .
- $R(x)b$: Variable x is read, the result is b .

Compulsory Exercise 5: Sequential Consistency

Submit via moodle!

Is the memory underlying the subsequent execution of two processes sequentially consistent? It is assumed that all variables are initially set to zero.

$P_1 : W(a)1; R(b)2; W(a)3$

$P_2 : R(a)1; R(a)3; W(b)2$