UNIVERSITÄT SIEGEN

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

Excercise Sheet 7

(To be processed until 05.12.)

Lecture Distributed Systems Winter Term 2024/25

Exercise 1: Clock Drift

a) Describe briefly a situation in which the clock drift in a distributed system can lead to problems.

b) Justify why leap seconds are inserted at certain intervals in the corrected UTC. If necessary, inform yourself about this via literature or the Internet.

Exercise 2: Clock Synchronisation

How can the clocks in two computers connected via a network be synchronized without using an external time source (e.g. GPS)? What can be said about the achievable accuracy? Which factors limit it?

Why is it impossible to realize perfectly synchronized clocks on two remote computers, even when using GPS?

Exercise 3: Programming: Clock Synchronization

In the archive `u07eFiles.zip`[1] on the lecture's web page you will find a simple class (`SynchronizedClock`) for the realization of an accurate clock (with time specification in ns), as well as a server that provides this time via RMI, and an associated client. The client checks whether its local clock is well enough synchronized with the server's clock by looking at three time stamps: at t_1 the RMI call takes place, at t_2 the processing in the server takes place, at t_3 the call returns in the client. One would expect here the sequence $t_1 \leq t_2 \leq t_3$. Since t_2 was measured with a clock other than t_1 and t_3, however, violations of this expectation are possible with unsynchronized clocks.

Realize the method `syncClock()` in `ClockClient`, so that the client synchronizes its local clock with that of the server. Since you cannot change the system clock, you may also have to extend the class `SynchronizedClock`.

Why is it possible that one minute after the clock synchronization the clocks do no longer match exactly enough?

Note: In order to be able to meaningfully solve this task on a single computer, the class `SystemClock`, which represents the system clock, is implemented in such a way that it adds a random offset to the time of the computer.

Exercise 4: 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 $8ms$.

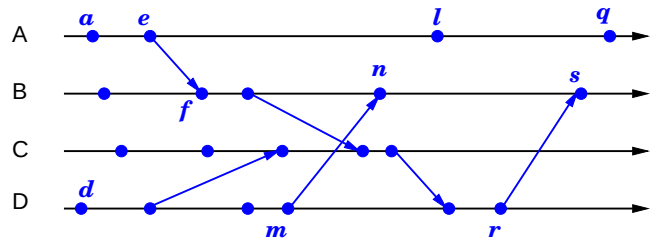
Sending time [hh:mm:ss]	Round-trip time [ms]	Server time [hh:mm:ss]
11:51:20.043	22	11:51:20.564
11:51:21.821	25	11:51:22.340
11:51:24.715	20	11:51:25.232

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u07eFiles.zip`

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 5: Lamport and Vector Time (Mandatory exercise for 6 CP, submit via moodle!)

A distributed computer system is given with four computers as well as the following sequence of receive, send and local events:



a) Specify pairs of events for which a causal order (in the sense of Lamport's *happened-before* relation) is not defined, i.e., which are concurrent. Can you make a statement as to whether the event d (actually) causally influenced the event n?

b) Now assume that the system realizes Lamport clocks for each computer. Enter the corresponding Lamport time-stamps for each event in the diagram above.

c) What can you infer from the Lamport timestamps of the events e and r?

d) Now enter the vector times for the events a, f, l, m and s.

e) What can you infer from the vector timestamps of the events l and m or from the vector timestamps of the events f and s?

Exercise 6: *Happened-Before* Relation, Lamport and Vector Time (Mandatory exercise for 6 CP, submit via moodle!)

Three processes A, B and C each generate the events $a_1, a_2, ..., b_1, b_2, ...$ and $c_1, c2, ...$

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 and vector time in threir relation to Lamport's *happended before* relation?

Exercise 7: Programming: Vector Clock

In the archive `u07eFiles.zip`[2] 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.

Complete the implementation of a vector clock specified in the file `Server.java` and extend the server so that the predefined method `log()` also outputs the current vector time for each event. The vector time should be increased with each receive attempt (call of `receive()`). The client is **not** to be considered for the vector time.

For simplification two methods `addToMessage()` and `extractFromMessage()` are given in the class `VectorClock`, which add the vector time to a string or extract it from it.

[2] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u07eFiles.zip`