TUM I 13
Chair of Application and Middleware Systems
Prof. Dr. H.- A. Jacobsen

Tutorial
Distributed Systems (IN2259)
WS 2020/21

Exercises on Time in Distributed Systems

Exercise 1 Cristian's Algorithm

A client attempts to synchronize its clock with a time server using Cristian's algorithm. Therefore, the client records the round-trip times of requests and the timestamps returned by the server as depicted in Table 1.1.

Round-trip time (ms)	Server time (hh:mm:ss:fff)
22	10:54:23:000
25	10:54:25:000
20	10:54:28:000

Table 1.1: Round-trip times and timestamps.

(a) Which of these times should the client use to adjust its clock?

(b) To what time should the client set its clock?

(c) Estimate the accuracy of the setting with respect to the server's clock.

(d) If it is known that the time between sending and receiving a message in the system is at least 8 ms, do your answers to the above questions change?

(e) Discuss if we can synchronize the client's clock with the time server to within 0 milliseconds, (i.e., fully accurately).

Exercise 2 Berkeley Algorithm

A collection of five processes (P1 - P5) want to synchronize their clocks according to the Berkeley algorithm. Process P3 has been elected as the master. The threshold value for acceptable deviation $\delta = \pm 3000ms$. For the given round of the algorithm, the times of all individual processes have been collected by the Master (cf. Table 2.2). (*Hint: Different to the original Berkeley algorithm, the master did not record any RTTs in this setting. So for this task, you can omit the calculation of a better estimate for the individual process times and instead work with the times given in the table.*)

Process	Process time (hh:mm:ss:fff)
P1	08:44:56:144
P2	08:44:52:874
P3	08:44:53:123
P4	08:44:53:100
P5	08:44:50:996

Table 2.2: Individual process times gathered at the master.

(a) What is the main difference between the Berkeley algorithm compared to Cristian's algorithm?

(b) What time will the master process P3 calculate as the reference time for synchronization.

(c) What information, i.e., which values will P3 send to the other processes for synchronization? What is the content of the message?

(d) Give an advantage and a disadvantage for sending, in particular, those kind of values that have been calculated in subtask (c) to synchronize the clocks.

EXERCISE 3 Timestamping with logical clocks

Consider the time-event diagram depicted in Figure 3.1 and accomplish the following tasks. Each arrow in the diagram represents a message transmission. For example, a is the sending event and b is the receiving event for a message that is sent by process A to process B.

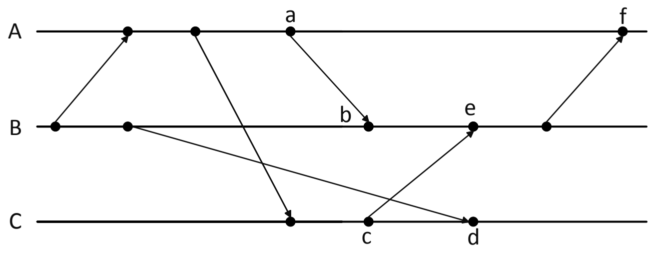

Figure 3.1: Time-event diagram.

(a) Timestamp each event based on the *Logical Clock* algorithm introduced by Lamport.

(b) Timestamp each event based on the *Vector Clock* algorithm.

(c) Using only the Vector Clock timestamps calculated in subtask (b) show whether, for each of the below event pairs, the *happened-before* relation ($\rightarrow$) holds or not. Two events that are not in *happened-before* relation are also called *concurrent* ($\|$) events.

(i) a, c

(ii) b, d

(iii) c, e

(iv) c, f

(d) Can we certainly determine the existence of either *happened-before* or *concurrency* relation between a pair of events only by knowing logical clock timestamps?

EXERCISE 4 Atomic total-order broadcast

Assume an asynchronous distributed system without failures that is comprised of n processes: $P = \{p_1, p_2, \ldots, p_n\}$. The only available communication mechanism in this system is broadcast, i.e., if a process sends a messages, this message is delivered to all processes in the system including the sender itself. Now, this system should be provided with a new feature referred to as *atomic total-order* broadcast (*ATO-broadcast*). In *ATO-broadcast* each process p_i ($1 \leq i \leq n$) receives messages in the same order as every other process.

For example, if process p_i receives messages in the order $[m_2, m_1, m_6 \ldots, m_k]$, then all other processes must receive messages in this order. We assume that processes are aware of their *processIds* (i.e., $\{p_1, p_2, \ldots, p_n\}$) and that no message gets lost in the communication between processes. Also, messages from the same sender are received in the order that they were sent (i.e., FIFO channel).

Design an algorithm to provide the system with this feature using *Lamport clocks* and process IDs. **Hint:** Write two pseudo code snippets. One describes the algorithm on the sender side (i.e., function `broadcastATO(msg)`), the other describes the algorithm on the receiver side (i.e., function `onReceiveATO(msg)`). In the pseudo code, you are allowed to use basic data structures like arrays, lists, queues, etc. with a simplified syntax.

Exercise 5 Causal Broadcast

Given the following definition and the logical clock algorithms discussed in the lecture, provide an algorithm to implement *Causally-ordered broadcast* (*CO-broadcast*). *CO-broadcast* extends the usual broadcast semantics with the following properties: (Note: $\rightarrow$ is the *happens-before relation*).

(i) Let m and m' be two broadcast messages such that broadcast(m) $\rightarrow$ broadcast(m'), then each process must receive m before m';

(ii) Let m and m' be two broadcast messages such that broadcast(m) $\|$ broadcast(m'), then m and m' can be received in different orders by different processes;