

# Dependable Distributed Systems

## Exercise week 2

October 6<sup>th</sup>, 2022

### Exercise 1

With reference to the synchronization of physical clocks, provide the definition of internal and external clock synchronization. In addition, consider a system composed of two processes  $p_1$  and  $p_2$  and one UTC server  $p_s$ . Let us assume that:

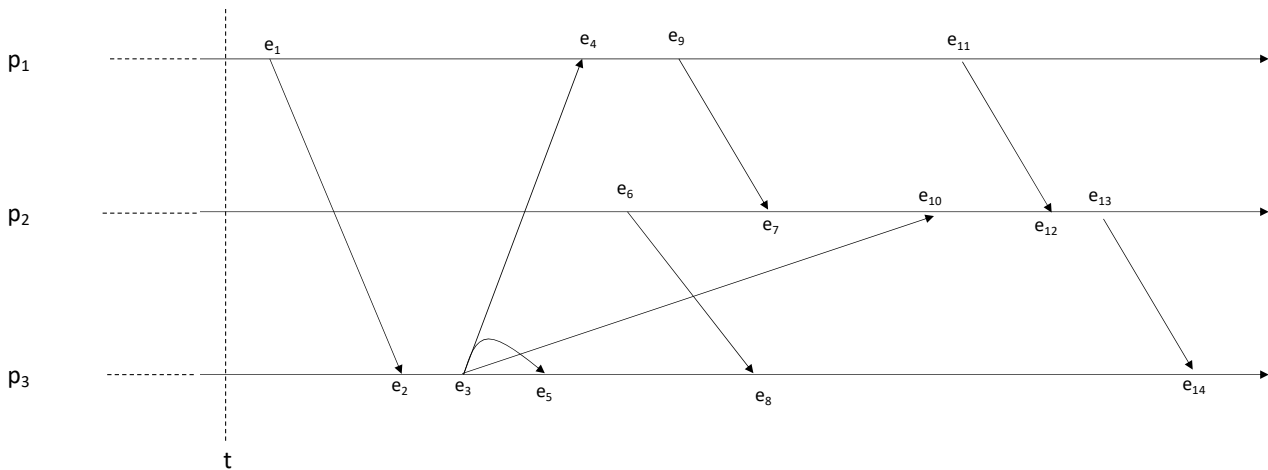
- $p_1$  and  $p_2$  communicate with  $p_s$  by using perfect point-to-point links
- the maximum latency of the channel between  $p_s$  and  $p_1$  is 1 ms
- the maximum latency of the channel between  $p_s$  and  $p_2$  is 2 ms.

In addition, let us assume that  $p_1$  and  $p_2$  start a clocks synchronization procedure at a certain time  $t$  by running the Cristian algorithm. Answer to the following questions:

1. how much is the accuracy bound  $D_{ext}$  of the external synchronization obtained by  $p_1$  and  $p_2$  at the end of the synchronization?
2. Is the current system internally synchronized? If yes, determine the internal synchronization bound  $D_{int}$  obtained at the end of the procedure.

### Exercise 2

Let us consider the execution history depicted in the figure

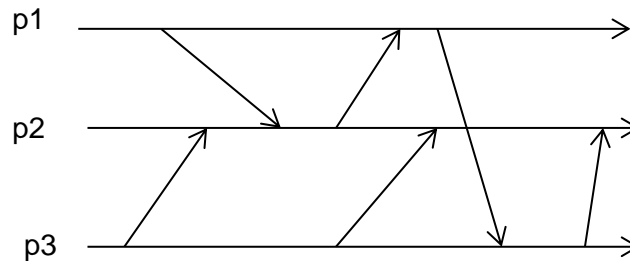


| Given the run depicted in the figure state the truthfulness of the following sentences: |   |   |   |
|---|---|---|---|
| a   | According to the happened-before relation, $e_5 \rightarrow e_7$  | T | F |
| b   | According to the happened-before relation, $e_4 \parallel e_5$  | T | F |
| c   | Let $CK_i$ be the variable storing the scalar logical clock of process $p_i$ . Let us assume that at time $t$ , $CK_i = 0$ for each process $p_i$ . The logical clock $CK_2$ associated to $e_6$ is strictly larger than the logical clock $CK_1$ associated to $e_1$ | T | F |
| d   | Let $CK_i$ be the variable storing the scalar logical clock of process $p_i$ . If at time $t$ $CK_1 = 0$ then the logical clock associated to $e_8$ is $CK_3=3$   | T | F |
| e   | Let $CK_i$ be the variable storing the vector logical clock of process $p_i$ . If at time $t$ $CK_1 = [0, 0, 0]$ then the logical clock associated to $e_8$ is $CK_3=[3, 0, 3]$   | T | F |

For each point, provide a justification for your answer

### Exercise 3

Describe timestamping techniques based on scalar logical clocks and vector logical clocks. In addition, considering the execution reported in Figure, answer to the following questions:



1. Apply the scalar clock timestamping technique to the execution assigning a timestamp to each event
2. Apply the vector clock timestamping technique to the execution assigning a timestamp to each event
3. List all pairs of concurrent events in the proposed execution

### Exercise 4

Consider an asynchronous message passing system that uses vectors clock to implement some causal consistency check. The message passing system is composed by 4 processes with IDs 1 to 4, and, as usual, the ID is used as displacement in the vector clock (i.e., the locations are (p1, p2, p3, p4) ).

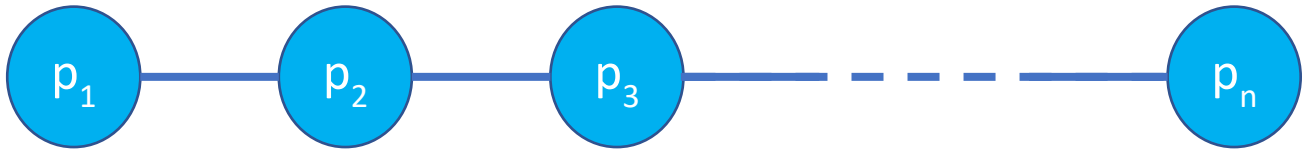
Vector clocks are updated increasing before send. Processes communicate by point2point links. You start debugging process p1 (the process with id 1) in the middle of the algorithm execution. You see the following stream of messages exiting and entering the ethernet card of process p1:

- Time 00:00 – EXITING: Send Message [MSG CONTENT] Vector Clock: (1, 0, 0,0)
- Time 00:05 – ENTERING: Rcvd Message [MSG CONTENT] Vector Clock: (1, 2, 3,1)

1. Draw an execution that justifies the vectors clocks you are seeing. Is such an execution unique?
2. There exists an execution that justifies the vector clocks and where there exists at least a process that does not send any message? Justify your answer.

## Exercise 5

Let us consider a distributed system composed of  $N$  processes  $p_1, p_2, \dots, p_n$  each one having a unique integer identifier. Processes are arranged in line topology as in the following figure



Let us assume that there are no failures in the system (i.e., processes are always correct) and that topology links are implemented through perfect point-to-point links.

Write the pseudo-code of a distributed algorithm that is able to build the abstraction of a perfect point-to-point link between any pair of processes (also between those that are not directly connected).