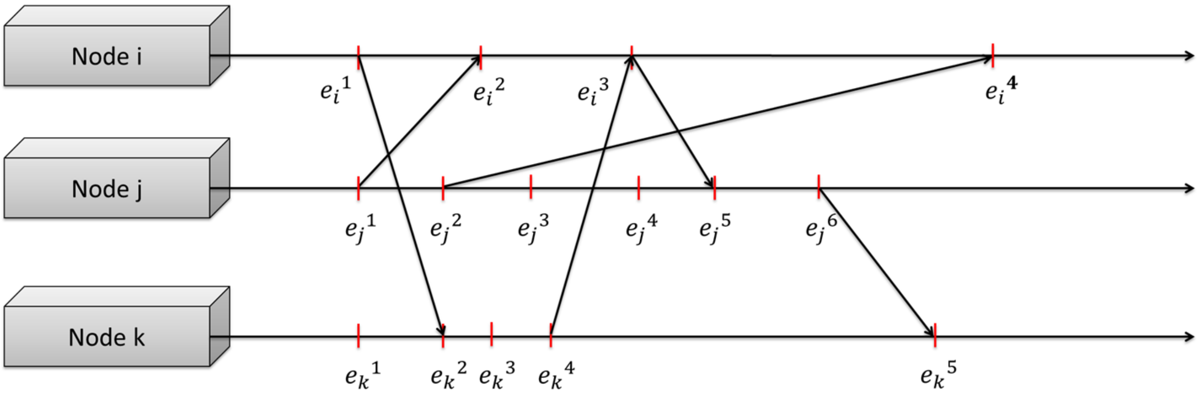
# Exam
# DT136G – Programming of Distributed Systems

13th of January 2021

Responsible teacher: Erik Schaffernicht

| | |
|---|---|
| **Aids:** | Prepared front and back page of a single A4 sheet. This ‘cheat sheet’ is to be submitted alongside the exam answers. |
| **Questions:** | Erik is available by phone (019 30) 3227 between 3 pm and 4 pm. |
| **Points:** | Each task has its point value listed, indicating how extensive the answer is expected to be. |
| **Comments:** | Read the tasks carefully and answer what the task asks for. Write legibly. Extended bullet points are fine as long as they clearly contain the arguments. There is no need to write an essay. There is neither partial nor total order in which the tasks have to be answered, but clearly indicate which answer refers to which task. |
| **Grading scale:** | ECTS (A-F) |

## Task 1 – Logical clocks (4 points)




Use logical clocks and the synchronization method suggested by Lamport to establish a *partial potential causal order (ppco)* for the events and messages (indicated by arrows) shown above.
[Note: The event $e_i^{\,3}$ is a single event that includes both a receive and a send operation.]

Below are given two pairs of events A & B. For each of them decide **(1)** whether *ppco*(A)<*ppco*(B), *ppco*(A)>*ppco*(B) or *ppco*(A)=*ppco*(B) is true, providing the actual numbers, and **(2)** whether A↪B, B↪A or A?B (to denote concurrent events in the "happens before" relationship ↪) is correct.

i) $e_k^{\,3}$ & $e_j^{\,4}$ ii) $e_j^{\,5}$ & $e_k^{\,3}$

## Task 2 – RAFT around the world (3 points)

Suppose that you implemented Raft and deployed it with all servers in the same datacenter. Now suppose that you were going to deploy the system with each server in a different datacenter, spread over the world. What changes would you need to make, if any, in the wide-area version of Raft compared to the single-datacenter version, and why?

## Task 3 – Distributed system properties (4 points)

The key requirements for the Google search engine infrastructure are scalability, reliability, performance, and openness. Provide three examples of where these requirements might conflict with each other and discuss how Google may deals with these potential conflicts.

## Task 4 – Caching (2 points)

How does caching help a name service’s availability? What is the 'price' to pay?

## Task 5 – Two-player match making (15 points)

Imagine a slow, turn-based online strategy game where two players play against each other (the actual game does not matter, but if it helps you can think of chess). Everything in the game is run by a centralized system with dedicated servers for the matches and one for the lobby / matchmaking.

After logging in, a player finds themselves in a lobby where they select the type of game they want to play. After making this selection the player is placed into a queue waiting to be matched against another player of similar skill level (measured by an arbitrary number stored by the matchmaking server). During this waiting time the player has the option to leave the queue by hitting the cancel button.

- If a player wants to start a match, the client informs the matchmaking server about that

- If there is another player of the same skill level waiting, server will match both players and send them a message containing the address of the server that will host the actual game to which the players will then connect to.
- If there is no other player of the same skill level, the server will inform the client that it must wait, and the client will be placed in a waiting queue.
- If a player is waiting to be matched, the player can cancel the matchmaking, resulting in the removal of the waiting status on the server side.
- If a player is waiting for longer than time *t* to get matched, the server will match that player with another player that is of a different skill level and send them a message containing the address of the server that will host the actual game to which the players will then connect to.

Assumption: There are no communication or node failures.

**A)** Write pseudocode for the matchmaking server and the clients modeling the above behavior through exchanging messages between nodes. [Clients and matchmaking server, do not include the actual game server!] Clearly indicate the use of asynchronous and synchronous messages. (6 points)

**B)** Using your solution described in the pseudocode above, draw a sequence diagram for the following situation:

No player is waiting for a match at the beginning.
$t_1$: Player A (Skill level: 3) signs up for a match.
$t_2$: Player B (Skill level: 2) signs up for a match.
$t_3$: Player B cancels their match sign up.
$t_4$: Player C (Skill level: 2) signs up for a game

$t_5$: The waiting threshold for player A has passed and they can now be matched without skill level restrictions.

The diagram ends when the server sends the players the game server addresses. (4 points)

**C)** Let us assume that the messages of $t_3$ and $t_4$ are received by the server in the inverse order (message from player C is received by the server before the message from player B is received). What is the experience of player B? Suggest how to fix this issue! (2 points)

**D)** Describe the behavior of your system in case

i) The matchmaking server fails.
ii) A player client fails.
iii) Messages are being lost.

All failures are considered fail-stop failures. (3 points)