University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

# Theory of Distributed Systems

Mock Exam - Summer Term 2025

Name: ..................................................................

Matriculation No.: ..................................................................

Signature: ..................................................................

## Do not open or turn until told so by the supervisor!

- Write your **name** and **matriculation number** on this page and **sign** the document.
- Your **signature** confirms that you have answered all exam questions yourself without any help, and that you have notified exam supervision of any interference.
- **No electronic devices** are allowed.
- Write legibly, use clear language, and only use a pen (ink or ball point). Do **not use red**! Do **not use a pencil**!
- You may write your answers in **English or German** language.
- Only **one solution per task** is considered! Make sure to strike out alternative solutions, otherwise the one yielding the minimal number of points is considered.
- **Detailed steps** might help you to get more points in case your final result is incorrect.
- The keywords **Show...**, **Prove...**, **Explain...** or **Argue...** indicate that you need to prove or explain your answer carefully and in sufficient detail.
- The keywords **Give...**, **State...** or **Describe...** indicate that you need to provide an answer solving the task at hand but without proof or deep explanation (except when stated otherwise).
- You may use information given in a **Hint** without further explanation.
- **Read each task thoroughly** and make sure you understand what is expected from you.
- **Raise your hand** if you have a question regarding the formulation of a task or if you need additional sheets of paper.
- A total of **45 points** is sufficient to pass and a total of **90 points** is sufficient for the best grade. Note that this mock exam contains about 20% more questions than a regular 2-hour exam.
- Write your name on **all sheets**!

| Task | 1 | 2 | 3 | 4 | 5 | 6 | **Total** |
|---|---|---|---|---|---|---|---|
| Maximum | 30 | 14 | 20 | 20 | 16 | 20 | 120 |
| Points | | | | | | | |

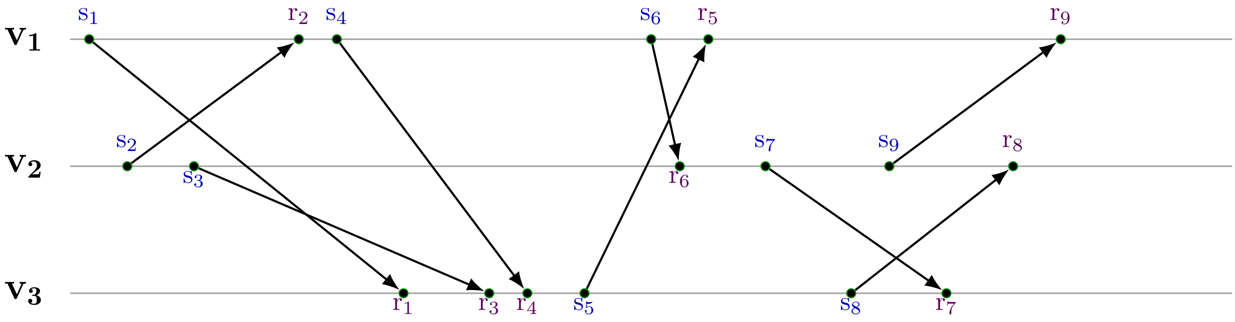
## Task 1: Multiple Choice Questions *(30 Points)*

Answer each of the following questions by marking either **True** or **False**. For each correct answer, you will get +2 points, for each wrong answer, you get −2 points, and whenever you do not answer a question, you get 0 points. If the total number of points for all the 15 questions below becomes negative, we round it up to 0.

| Question | True | False |
|---|---|---|
| In an asynchronous network with a leader node $v$, one can use the “flooding & echo” algorithm from the lecture to determine the number of nodes of the network in time $O(n)$. | ☐ | ☐ |
| In asynchronous networks, the message complexity of the “flooding & echo” algorithm from the lecture is asymptotically larger than the message complexity of the same algorithm in synchronous networks. | ☐ | ☐ |
| The distributed Bellman-Ford algorithm that was discussed in the lecture can be used to compute a BFS tree with a worst-case message complexity of $O(n^2)$. | ☐ | ☐ |
| The agreement property of the consensus problem is a safety property. | ☐ | ☐ |
| The validity property of the consensus problem is a liveness property. | ☐ | ☐ |
| A reduction discussed in the lecture shows that if a randomized $O(\sqrt{\log n})$-round algorithm for computing a $(\Delta+1)$-coloring exists, then such an algorithm also exists for computing an MIS. | ☐ | ☐ |
| Consider a synchronous fully connected network in which nodes can experience crash failures. Assume that whenever a node fails in some round $r$, then either all or none of the round-$r$ messages of the node are delivered. In this case, there is an $f$-resilient consensus algorithm that runs in $\leq \lceil f/2 \rceil$ rounds. | ☐ | ☐ |
| In a fully connected synchronous network with 3 nodes in which messages can get lost, there is no deterministic consensus algorithm. | ☐ | ☐ |
| Consider the family of $n$-node graphs with $O(n)$ edges. For this family of graphs, there exists an asynchronous, deterministic leader election algorithm with a message complexity of $O(n \cdot \sqrt{\log n})$. | ☐ | ☐ |
| In graphs of maximum degree 10, there exists a synchronizer $\mathcal{S}$ with time complexity $T(\mathcal{S}) = O(1)$ and message complexity $M(\mathcal{S}) = O(n)$. | ☐ | ☐ |
| The nodes of the *similarity graph* are all the possible executions of a given distributed system. Two nodes are connected by an edge in the graph iff they are similar (as defined in the lecture). Fix some $R \geq 1$. The similarity graph of the set of all possible $R$-round executions of the two generals problem is connected. | ☐ | ☐ |
| When applying the Arrow protocol on a balanced binary tree $T$, the time complexity of each request is at most $O(\log n)$. | ☐ | ☐ |
| The optimistic price of anarchy of a game is always smaller than the price of anarchy of the game. | ☐ | ☐ |
| Consider a shared-memory system consisting of (sufficiently many) read-write and test-and-set registers. In such a system, there is an asynchronous wait-free deterministic algorithm to solve consensus with three processors. | ☐ | ☐ |
| With the Cole-Vishkin algorithm that was discussed in class, one can color unrooted $n$-node trees with 6 colors in $O(\log^* n)$ rounds. | ☐ | ☐ |

## Task 2: Schedule and Clocks *(14 Points)*

Consider the following Schedule $S$:




(a) Which of the following "Happens-Before" relations hold? No argumentation required. *(4 Points)*

- $s_1 \Rightarrow_S s_3$
- $s_2 \Rightarrow_S r_3$
- $s_4 \Rightarrow_S s_9$
- $r_1 \Rightarrow_S s_9$

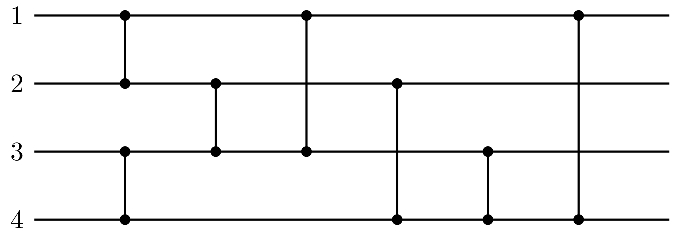
(b) Given some schedule $S' = s_1 s_2 r_2 s_4 r_1 s_3 r_3 s_6 r_6 r_4 s_5 r_5 s_7 s_9 r_9 s_8 r_7 r_8$, where the 9 messages are sent and received by the same processes as in the above schedule $S$. Give the local views $S'|v_i$ for $i = 1, 2, 3$. *(4 Points)*

(c) Is $S'$ a Causal Shuffle of $S$? Justify your answer. *(2 Points)*

(d) Apply the Lamports Clocks algorithm to $S$. Make sure you state the clock values $\tau(e)$ for all events $e$ in $S$ (for simplicity you can write the values into the figure above). *(4 Points)*

## Solution Task 2

## Task 3: Sorting Networks *(20 Points)*




(a) Is the above sorting network correct? If so, argue why, otherwise give an example input that will not be sorted correctly. *(4 Points)*

(b) Consider an arbitrary sorting network with 4 wires. Why is it sufficient to only check 11 possible inputs to validate the correctness of this sorting network for any arbitrary input in $\mathbb{Z}^4$? *(4 Points)*
*Hint: First show that already sorted input sequences will always form a sorted output sequence.*

(c) Consider a correct sorting network with $n$ wires. Show that for each pair of adjacent wires $i, i+1$, the network contains a comparator that compares wires $i$ and $i+1$. *(4 Points)*

(d) Is the following statement true or false: Given any correct sorting network, adding another comparator anywhere does not destroy the sorting property. Explain your solution. *(8 Points)*

## Solution Task 3

## Task 4: Distributed Edge Coloring *(20 Points)*

A proper edge coloring is an assignment of colors (i.e., numbers) to the edges, such that no two edges that share a node have the same color, e.g., if there are the edges $e_1 = \{u, v\}, e_2 = \{v, w\}, e_3 = \{w, x\}$, one can not color $e_1$ and $e_2$ with the same color (as they share node $v$), also $e_2$ and $e_3$ cannot be colored with the same color while $e_1$ and $e_3$ do not violate this property if they have the same color. In the following assume a synchronous message passing system on a graph $G$ with $n$ nodes, $m$ edges and maximum degree $\Delta$. We assume that each node knows these 3 parameters, and we further assume that all nodes have unique IDs $\in \{1, ..., n\}$. Consider the following algorithm:

**Algorithm 1** Simple Edge Coloring

1: Each node $v$ assigns unique values in $\{0, ..., \Delta-1\}$ to its adjacent edges. Thus, each edge $e = \{u, v\}$ is assigned two numbers $x_{u,e}$ and $x_{v,e}$.
2: Each node sends those values to its neighbors.
3: Assume $id(u) < id(v)$: Color edge $e = \{u, v\}$ with color $c(e) := \Delta \cdot x_{u,e} + x_{v,e}$

(a) What is the range of edge colors assigned by this algorithm? More specifically, what are the values of $c_{min}$ and $c_{max}$ such that $c(e) \in \{c_{min}, ..., c_{max}\}$ for all edges $e$. *(2 Points)*

(b) What is the time complexity of the above algorithm. *(2 Points)*

(c) Does this algorithm produce a proper edge coloring? If so, give a formal proof that no two adjacent edges can have the same color. Otherwise, give an example where the algorithm assigns the same color to two edges of some node $v$. Also what is the maximum number of incident edges of a given color that a node $v$ can have. *(8 Points)*

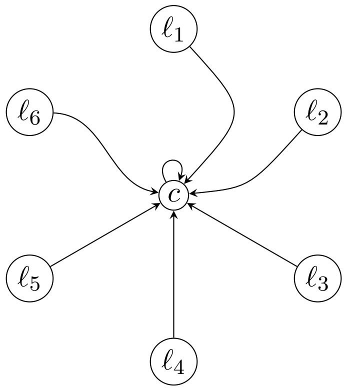
(d) We now aim for a proper edge coloring. Give an $O(\log^* n)$-round (deterministic) algorithm that colors the edges with at most $O(9^{\Delta})$ colors. *(8 Points)*

## Solution Task 4

## Task 5: Shared Objects *(16 Points)*

In the following, we consider the Ivy protocol that was discussed in the class for handling a shared object in a fully connected network. Recall that the Ivy protocol works essentially in the same way as the Arrow protocol, however when answering a request, the protocol uses path contraction to adjust the topology of the used spanning tree.

Assume that we have a system consisting of nodes $c$ and $\ell_1, \ldots, \ell_k$ for some integer $k$. The initial configuration is a star, where node $c$ is in the center and every node is pointing towards $c$ (see figure below). Give a sequence of requests such that, when applying the Ivy protocol, the final configuration is a path with $c$ at one end and with everything directed towards $c$. Explain why your sequence of requests results in a path.




**Hint:** Think about what happens when you make a few consecutive requests at nodes directly pointing towards $c$.

## Solution Task 5

## Task 6: Game Theory *(20 Points)*

Alice and Carol went on a trip and bought two identical antique vases for 500 Euros each. Unfortunately, their luggage was not handled properly when flying back home, and both vases broke. The airport manager wants to compensate Alice and Carol for the broken vases. As he does not know the true price of such a vase, he proposes a scheme.
Alice and Carol are placed in separate rooms so that they cannot communicate. Each of them is asked to write down the price of one vase as a natural number between 20 and 1000. Then, let $p_{Alice}$ and $p_{Carol}$ denote the prices they wrote and let $\Delta$ denote an integer ($0 \leq \Delta \leq 20$). They will be compensated as follows:

- If $p_{Alice} = p_{Carol} = p$, then $p$ must be the true price of the vase and Alice and Carol receive $p$ Euros each.
- If $p_{Alice} > p_{Carol}$, the airport manager assumes that Alice lied and Carol's price is correct. Then, Alice receives $p_{Carol} - \Delta$ Euros, and Carol receives $p_{Carol} + \Delta$ Euros.
- Symmetrically, if $p_{Alice} < p_{Carol}$, the airport manager assumes that Carol lied and Alice's price is correct. Then, Alice receives $p_{Alice} + \Delta$ Euros, and Carol receives $p_{Alice} - \Delta$ Euros.

Alice and Carol are both smart and rational. Each of them wants to maximize their own compensation, even if it means to lie to the airport manager.

(a) Is writing down the true price of the vase a dominant strategy for Alice when $\Delta = 5$? Why? *(4 Points)*

(b) Find a pure Nash Equilibrium for $\Delta = 5$. *(8 Points)*

(c) For which values of $\Delta$ is there a unique pure Nash equilibrium? Justify your answer. *(8 Points)*

## Solution Task 6