Delft University of Technology - EEMCS

TU Delft

Exam Distributed Algorithms (IN4150)

3 Feb 2023

Please read the following information carefully! (the same information is on the exam and on the answer sheet)

- Opening the exam and/or answer sheet before you are instructed to start is **strictly prohibited**.
- This exam consists of 15 questions/exercises. The number of points each question is worth is written next to the question (the complete exam is **73** points).
- You have **180 minutes** to complete this exam.
- Before you hand in your answers, check that your answer form contains your name and student number on every page, also filled in using the boxes.
- Please check carefully which question should be answered where and stick to the indicated space.
- The use of books, notes, calculators, smart watches, and other aids is **strictly prohibited**.
- Any answer that you provide to an open question should be backed up by an explanation and/or the accompanying maths!
- Fill in the answer sheet with blue/black **pen**. If you make a mistake on the answer form, there is extra space at the end. If you use this extra space: **indicate this clearly in the original question box that your answer is or continues in the extra space.**
- You can solve the exercises in any order, and the order of the exercises is **not** determined by their difficulty. Go through the exam once before starting.

Independent questions [29 points].

1. (1 point) Describe how one can implement a virtual FIFO point-to-point channel over a reliable communication channel that might reorder or delay messages.

2. (2 points) Explain the difference between a partially synchronous and a fully asynchronous network.

3. (2 points) Give two distributed computing problems that have been shown to be impossible.

4. (2 points) Explain the difference between a fault and a failure.

5. (3 points) Give the definition of three standard fault models for processes.

6. (4 points) Explain how Awerbuch's gamma synchronizer organizes processes in groups of processes and combines synchronizers alpha and beta. Give a level description of the alpha and beta synchronizers, and explain how both interact in the gamma synchronizer.

7. (2 points) Explain the difference between receiving and delivering a message in the context of a total order broadcast protocol.

8. (2 points) Bob only knows the first and the last blocks of the Bitcoin blockchain, and he asking Alice to send him the blocks he is missing. Explain why it is not possible for Alice to tamper with the blockchain when answering Bob's request.

9. (2 points) Explain a situation where one might want to compute a minimum spanning tree.

10. (3 points) Explain the difference between the agreement and the consensus problems, and explain which of these two problems is the most difficult.

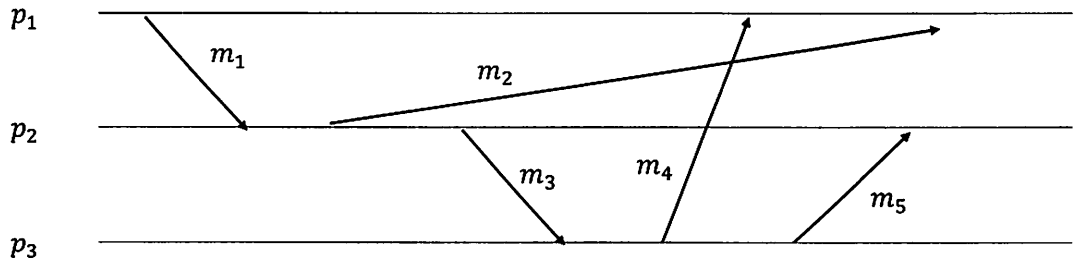
11. (2 points) Explain how the election problem can be solved by first building the minimum spanning tree of the network.

12. (2 points) Give an example of an actual system where causal order would be an important requirement. Explain why.

13. (2 points) In the context of self-stabilization, explain what it means for a process to be enabled.

14. **Vector clocks [17 points].**

Processes p_1, p_2 and p_3 communicate using a protocol that relies on vector clocks. Initially, all processes begin with their vector clock set to all zeros. Processes then exchange messages m_1, m_2, m_3, m_4 and m_5 as illustrated below.



(a) (2 points) List all events in the diagram that happen at processes p_1, p_2 or p_3 that are relevant to compute the value of vector clocks.

(b) (4 points) Indicate the clock value that is sent along with each message and the final clock value of each process.

(c) (3 points) Name a pair of causally connected events, and justify it using their vector clocks.

(d) (3 points) Name a pair of concurrent events, and justify it using their vector clocks.

(e) (3 points) (i) It is sometimes possible to detect that two events are concurrent using scalar clocks. Explain when. (ii) When it is not possible to decide whether two events are concurrent using scalar clocks, explain whether using vector clocks would help.

(f) (2 points) Indicate at what times processes would deliver each message if they were running a causal point-to-point algorithm.

15. **Building a virtual ring [27 points].**

Let us assume a system of N processes interconnected by a network. Each process knows its (unique) ID and the ID of all processes in the system. Processes are interconnected with authenticated channels and they know the ID of the process at the other end of their channels. We note $\Gamma(i)$ the neighbors of process i. The processes might be running various algorithms/systems concurrently. In this exercise, we study the problem of building a virtual ring.

(a) (5 points) We first focus on building a **(virtual) random unidirectional ring** assuming that the **network is fully-connected**. At the end of the algorithm each process should know the ID of its predecessor on the ring. The algorithm should not be determined by any particular process, and should return a different result each time it is executed (with some probability and if topologically-possible).

Your first task is to design a novel **token-based** algorithm that builds a **unidirectional ring**. You can assume that one particular process P_s receives a signal to start the algorithm (i.e., executes a function **initiate_ring()**), and that all processes execute a procedure **handle_msg(Msg m)** whenever they receive a message during the ring computation. Define and justify the format of the messages that processes exchange. Write the pseudocode of functions **initiate_ring()**and **handle_msg(Msg m)**.

(b) (4 points) Let us assume that all message fields are encoded over 32 bits. Give and justify the time complexity, the message complexity, and the bit complexity (total number of bits transmitted) of your algorithm. Explain whether this algorithm scales well with the number of processes.

(c) (2 points) Explain how to modify your algorithm so that it builds a bidirectional ring (i.e., processes know their predecessor and successor on the ring) instead of a unidirectional ring. Detail the impact of your modification(s) on the complexities you obtained in 2.(b).

(d) (2 points) Explain and justify whether your algorithm from 2.(a) allows all processes to detect its termination. If yes, explain why. If not, describe how you could modify your algorithm to ensure this property.

(e) (6 points) Describe a variant of your algorithm from 2.(a) that does not use a token, and only relies on messages that contain (at most) a message type and the ID of the process that started the algorithm. How many message types are necessary for your algorithm? Describe the worst-case execution of this algorithm and its associated complexities. Compare these worst-case complexities to the ones you obtained in 2.(b).

(f) (3 points) We now assume that the network is partially connected. Explain the conditions (i) on the network topology and (ii) on the actual execution for algorithm 2.(a) (or 2.(b)) to possibly terminate correctly. Explain how a process could verify whether this network topology condition is satisfied (e.g., by adapting one of the algorithm of the course).

(g) (3 points) Explain how to design a unidirectional ring construction algorithm, based on algorithm(s) from the course, that will **always** return a valid result in a partially-connected network.

(h) (2 points) Explain what bad consequences could happen with algorithm 2.(a) when several processes receive a signal to run **initiate_ring**. Explain how to prevent this from happening either with an algorithm from the course or with your own algorithm.

End of exam