Teacher: Professor Rachid Guerraoui
Exam: CS-451 - Distributed Algorithms
Exam date: December 19th, 2022

X

X

SCIPER : **X**

Signature :

Time Limit 3:00 hours.

Instructions

- This exam is closed-book: no notes, electronics, nor cheat sheets are allowed.
- Read through each problem before starting to solve it.
- Whenever presenting an algorithm, be as precise as possible, and preferentially use the pseudo-code notation used in class.
- When solving a problem, do not assume any known result from the lectures, unless it is explicitly stated that you might use some known result.

Important (read with care!)

Unless a problem explicitly states otherwise:

- Assume that communication links are **perfect**.
- Assume that **all** processes can fail.
- Assume that processes only fail by **crashing**.
- **Do not** assume that a **Failure Detector** is available.

The maximum score for this exam is 7. The grade for this exam is equal to the score achieved, capped at 6.
This means that a perfect score is not necessary to achieve the maximum grade!

Good luck!

Complexity & Latency Let `A` be an algorithm. We define the following:

- The **maximum per-process message complexity** of `A` is the maximum number of messages any process has to **send** throughout an execution of `A`.
- The **latency** of `A` is the number of units of time `A` takes to terminate (i.e., provide an output) on every correct process, under the following assumptions:
  - Computation is instantaneous.
  - Every message is delivered exactly one unit of time after being sent.

  Example. Consider an algorithm `A` running on three processes, `a`, `b`, and `c`. Upon initialisation, `a` sends a message `m` to `b`. Upon delivering `m`, `b` sends a message `m′` to both `a` and `c` and terminates. Upon delivering `m′`, both `a` and `c` terminate. The latency of `A` is 2. Indeed: `b` terminates after one unit of time (delivery of `m`); `a` and `c` terminate after an additional unit of time (delivery of `m′`).

---

a. *(0.25 points)* State the interface and properties of Regular Reliable Broadcast (RRB).

b. *(0.5 points)* Consider a single instance of RRB such that exactly one message can be sent by a single designated sender. Assume the sender cannot fail (all other processes can fail). Present an algorithm that implements RRB with the smallest possible latency. Prove the correctness of your algorithm and that the achieved latency is indeed the smallest possible.

c. *(0.25 points)* Consider the Knock-Off Regular Reliable Broadcast (KORRB) abstraction, defined as follows:

- The interfaces of KORRB and RRB are identical.
- The properties of KORRB are the same as RRB, minus Validity.

Consider a single instance of KORRB such that exactly one message can be sent by a single designated sender. Present an algorithm that implements KORRB with the smallest possible maximum per-process message complexity. Prove the correctness of your algorithm and that the achieved maximum per-process message complexity is indeed the smallest possible.

d. *(0.25 points)* State the interface and properties of Uniform Reliable Broadcast (URB).

e. *(0.25 points)* Implement RRB using URB. Prove the correctness of your algorithm.

Please answer inside the box!

Question 2 *Failure Detectors*

a. *(0.25 points)* State the interface and properties of Perfect Failure Detector ($\mathcal{P}$).

b. *(0.25 points)* Consider the Knock-Off Failure Detector ($KO - \mathcal{P}$) abstraction, defined as follows:

- The interfaces of $KO - \mathcal{P}$ and $\mathcal{P}$ are identical.
- The properties of $KO - \mathcal{P}$ are the same as $\mathcal{P}$, minus Accuracy.

Implement $KO - \mathcal{P}$ with the smallest possible maximum per-process message complexity (for the definition of the maximum per-process message complexity of an algorithm, see Question 1). Prove the correctness of your algorithm and that the achieved maximum per-process message complexity is indeed the smallest possible.

c. *(0.25 points)* State the interface and properties of Eventually Perfect Failure Detector ($\diamond\mathcal{P}$).

Please answer inside the box!

Question 3 *Uniform Consensus and Uniform Terminating Reliable Broadcast*

a. *(0.25 points)* State the interface and properties of Uniform Consensus (UC).

b. *(0.5 points)* Present an algorithm that implements UC using Perfect Failure Detector. Prove the correctness of your algorithm.

c. *(0.5 points)* Can UC among N processes be implemented using an Eventually Perfect Failure Detector under the assumption that $\lceil N/2 \rceil$ (or more) processes can fail?

- If yes, present an algorithm that does that. Prove the correctness of your algorithm.
- If no, prove that doing that is impossible.

d. *(0.25 points)* State the interface and properties of Uniform Terminating Reliable Broadcast (UTRB).

e. *(0.5 points)* Prove that UTRB cannot be implemented without Perfect Failure Detector.

f. *(0.25 points)* Consider the Rarely Terminating Uniform Reliable Broadcast (RTURB) abstraction, defined as follows:

- The interfaces of RTURB and UTRB are identical.
- The properties of RTURB are the same as UTRB, except Termination is exchanged for *Rare Termination*: If all processes are correct, then every correct process eventually delivers a message.

Can RTURB be implemented **without** Perfect Failure Detector?

- If yes, present an algorithm that does that. Prove the correctness of your algorithm.
- If no, prove that doing that is impossible.

Please answer inside the box!

Question 4 *Weak Validity Uniform Consensus and Strong Validity Uniform Consensus*

Weak Validity Uniform Consensus The Weak Validity Uniform Consensus (WUC) abstraction is defined as follows:

- The interfaces of WUC and UC are identical.
- The properties of WUC are the same as UC, except Validity is exchanged for *Weak Validity*: If all processes are correct and all processes propose the same value, then only that value can be decided.

Strong Validity Uniform Consensus The Strong Validity Uniform Consensus (SUC) abstraction is defined as follows:

- The interfaces of SUC and UC are identical.
- The properties of SUC are the same as UC, except Validity is exchanged for *Strong Validity*: A decided value is proposed by a correct process.

---

a. *(0.75 points)* Let `A` be an algorithm that implements WUC among N processes using an Eventually Perfect Failure Detector under the assumption that at most f processes can fail, where $N = 2f + 1$. Consider the following execution of `A`:

   - All faulty processes crash immediately (i.e., at time 0); faulty processes do not send any message before crashing.
   - All correct processes propose the same value v in the execution.

   Prove that all correct processes decide v in the execution.

b. *(0.75 points)* Prove that it is impossible to implement SUC among N processes using an Eventually Perfect Failure Detector under the following assumptions:

   - Up to f processes can crash, where $N = 2f + 1$.
   - Every process proposes either 0 or 1.

c. *(1 point)* Implement SUC among N processes using an Eventually Perfect Failure Detector under the following assumptions:

   - At most f processes can crash, where $N = 3f + 1$.
   - Every process proposes either 0 or 1.

   Prove the correctness of your algorithm.

Please answer inside the box!