Institute of Computer Science, University of Bern
Distributed Algorithms, Spring 2020

Prof. Christian Cachin
Jovana Mićić

Exercise 5

5.1 Regular register executions (4pt)

Figure 1 shows an example of a regular register execution. Process p does two *write* operations and process q does three *read* operations, where last two overlap with the *write* operation of p. Your task is to describe (or draw) the execution steps of

a) Algorithm 4.1: Read-One Write-All

b) Algorithm 4.2: Majority Voting Regular Register

of [CGR11] with respect to the scenario described in Figure1. Consider a system of no less than three processes.

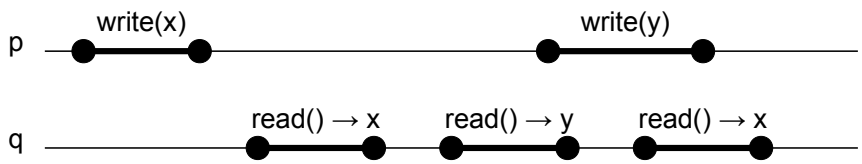

Figure 1. A regular register execution.

5.2 Read-all write-one regular register (3pt)

Implement an algorithm providing a $(1, N)$ regular register. Your protocol should be similar to Algorithm 4.1 [CGR11, Sec. 4.2.2]), but instead follow a *read-all write-one* approach. It should use the *fail-stop model*, where a perfect failure detector is available. When a process crashes, the failure detector ensures that eventually all correct processes detect the crash (*strong completeness*), and no process is detected to have crashed until it has really crashed (*strong accuracy*).

5.3 $(1, 1)$ Atomic register (3pt)

Study Algorithm 4.3 [CGR11, Sec. 4.3.2], which is an abstract transformation from one $(1, N)$ regular register to an $(1, 1)$ atomic register. Use the underlying idea to describe modifications for Algorithm 4.2 ("Majority voting regular register") [CGR11, Sec. 4.2.3] such that the modified protocol implements a $(1, 1)$ atomic register in the fail-silent model (where less than $N/2$ processes may fail).

You might be thinking that a $(1, 1)$ register is not very useful because the two clients (reader and writer) could simply communicate with each other. However, these clients may not be online simultaneously and the approach gives insight into more complex protocols.

References

[CGR11] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to reliable and secure distributed programming (Second Edition)*, Springer, 2011.