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

Prof. Christian Cachin
Jovana Mićić

Exercise 3

3.1 Relations among failure detectors (4pt)

A perfect failure detector abstraction is specified in Module 2.6 [CGR11]. Consider a *non-perfect* failure detector notion with the same interface as in Module 2.6, which only satisfies the following completeness notion:

> **NPFD:** *Weak completeness:* Eventually every process that crashes is permanently detected by *some* correct process.

But it still satisfies the *strong accuracy* property.

Implement a perfect failure detector abstraction from this non-perfect variant. (You should describe an emulation that satisfies all properties of the perfect failure detector; it may use the output from the non-perfect failure detector and send point-to-point messages.)

3.2 Perfect Failure Detector (4pt)

Assume that a bound Δ on message delay is known and that all local processing after delivering a point-to-point message takes at most Φ time. Algorithm 2.5 [CGR11, p. 51] implements a *perfect failure detector* with request/reply messages. Your task is to:

a) Design an alternative implementation that uses only unidirectional messages (that is, only periodic heartbeats from every process to all others instead of request/reply pairs between every pair of processes).

b) Discuss your algorithm in relation to Algorithm 2.5.

3.3 Quorum systems (2pt)

Quorum systems represent a fundamental abstraction for coordination among the nodes of a distributed system.

> **Quorum system:** Let $\Pi = \{p, q, \dots\}$ be a universe with n elements (e.g., processes). A *quorum system* $\mathcal{Q} \subset 2^{\Pi}$ is a set of subsets of Π such that every two subsets intersect. Each $Q \in \mathcal{Q}$ is called a *quorum*.

W.l.o.g., the quorum systems considered here are minimal, i.e., for any $Q, Q' \in \mathcal{Q} : Q \not\subseteq Q'$. Three example quorum systems are:

Singleton: $\mathcal{Q} = \big\{\{p\}\big\}$ for $n = 1$.

Majority: $\mathcal{Q} = \big\{Q \subset \Pi \;\big|\; |Q| = \lceil \frac{n+1}{2} \rceil\big\}$.

Grid: Suppose $n = k \cdot k$ and arrange the processes in a square. Every subset of Π is a quorum if and only if it consists of one full row and one element from each row below the full row.

$\Longrightarrow$

Suppose a protocol run by processes in Π needs some quorum of *correct* processes to operate, the remaining ones may *fail*. What are the minimum and maximum numbers of faulty processes tolerated by each of the three quorum systems?

Bonus question (+2pt): Suppose a client chooses randomly and with uniform distribution a quorum $Q \in \mathcal{Q}$ and sends a request to all processes in Q. Every $p \in \Pi$ therefore receives a request with a certain probability; this probability has been called the *load* of p under the quorum system. What is the maximal load on any process for each of the three quorum systems?

References

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