Prof. Christian Cachin
Jovana Mićić

# Exercise 6

## 6.1 Atomic register execution (5pt)

Study Algorithm 4.9 [CGR11, Sec. 4.4.4] for an $(N, N)$ atomic register in the fail-stop model (with a perfect failure detector $\mathcal{P}$). It illustrates how timestamp/process identifiers are used for linearizing the write operations.

Describe (or draw) two executions, $A$ and $B$, of this protocol with five processes $p$, $q$, $r$, $s$, and $t$. Decide on an order among $\{p, q, r, s, t\}$. Initially the register stores a value $\perp$. Process $p$ starts operation $\mathit{write}_p(x)$ at the same time as process $q$ starts $\mathit{write}_q(y)$. Processes $r$ and $s$ execute $\mathit{read}_r()$ and $\mathit{read}_s()$; both of these operations are concurrent to the two *writes*. Process $t$ executes $\mathit{read}_t()$ such that both *writes* precede this operation.

a) In execution $A$, we observe $\mathit{read}_r() \to x$ and $\mathit{read}_s() \to y$.

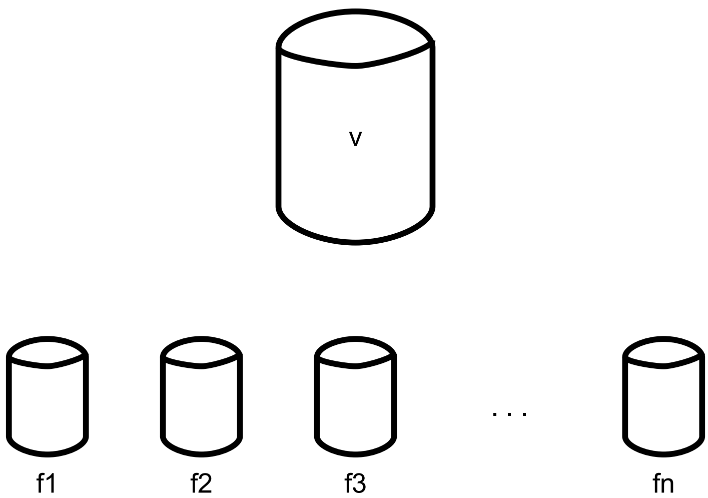
b) In execution $B$, we observe $\mathit{read}_r() \to y$ and $\mathit{read}_s() \to x$.

What does $\mathit{read}_t()$ return in $A$ and in $B$?

## 6.2 Erasure-coded storage (5pt)

Practical distributed storage systems often use *erasure coding* to reduce the space taken by redundantly stored data.

An $(k, n)$-*erasure code* maps a large data value (such as a disk sector or a file, which consists of $k$ information blocks) to $n$ so-called *fragments* of smaller size. The erasure code ensures that an encoded value can be reconstructed from any $k$ fragments (or code blocks). Practical erasure codes are based on Reed-Solomon codes over finite fields, for example.




$\Longrightarrow$

More precisely, an $(k, n)$-erasure code with domain $\mathcal{V}$ is given by two algorithms *encode* and *decode*:

- Algorithm $\mathit{encode}_{k,n}(v)$, when given a (large) value $v \in \mathcal{V}$, produces a vector $[f_1, \ldots, f_n]$ of $n$ fragments, with $f_i \in \mathcal{F}$. A fragment is typically much smaller than the input, and any $k$ fragments contain all information of $v$, that is, $|\mathcal{V}| \approx k|\mathcal{F}|$. (In other words, we consider only *maximum-distance separable codes* here.)
- For an $n$-vector $F \in \big(\mathcal{F} \cup \{\bot\}\big)^n$, whose entries are either fragments or the symbol $\bot$, algorithm $\mathit{reconstruct}_{k,n}(F)$ outputs a value $v \in \mathcal{V}$ or $\bot$. Output $\bot$ means that the reconstruction failed. In other words, if one computes $F \leftarrow \mathit{encode}_{k,n}(v)$ for some $v \in \mathcal{V}$ and then erases up to $n - k$ entries in $F$ by setting them to $\bot$, algorithm $\mathit{reconstruct}_{k,n}(F)$ outputs $v$. Otherwise, the algorithm outputs $\bot$.

The replication method of the storage protocols considered in the class can be seen as a $(1, n)$-erasure code. The RAID-5 encoding scheme found in practical disk controllers corresponds to an $(n - 1, n)$-erasure codes, which can be implemented solely by XOR operations. (Analogously, RAID-6 implements an $(n - 2, n)$-erasure code.)
Tasks:

a) Consider a distributed storage system of $n$ nodes, of which $f < n/2$ may fail by crashing. Pick a suitable erasure code and describe the *storage efficiency* of the system, i.e., the ratio of stored information and provisioned space that must be provisioned.

b) Modify the majority-voting protocol implementing a regular register (Algorithm 4.2 [CGR11]) to implement an erasure-coded *safe* register.

c) Why is it difficult to extend this protocol to regular semantics?

# References

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