Exercise 10: Consensus

Task 1: More Value(s)

In this task, we have a look at *multivalued consensus,* which is like binary consensus, except that inputs and outputs are from a larger domain $[O]$ for some $2 < O \in \mathbb{N}$. Our goal is an efficient reduction to binary consensus.

a) Give a 1-round algorithm with the following property. If all correct nodes have the same input, each node outputs this value. Otherwise, there is a single value $o \in [O]$ such that all nodes output either o or 0.

b) We replace all inputs with the output computed in a). Then we use binary consensus to decide whether to use output 0 or output o. What is the problem with this naive approach?

c) Fix the problem from b) in a way that still guarantees that if *all* correct nodes had opinion o after a), then the binary consensus routine will decide in favor of o. Exploit that if this is not the case, it is perfectly fine to output 0!

d) Plug these results together to obtain the desired reduction. Show that it costs $\mathcal{O}(1)$ rounds with messages of size $\mathcal{O}(\log O)$.

e*) If $f \ll n$, this reduction costs $\omega(fn)$ messages. Can you adapt the solution so that $\mathcal{O}(fn)$ messages (and $\mathcal{O}(fn \log O)$ bits) are sent by correct nodes in total?

Solution

a) Each node broadcasts its input. If it receives at least $n - f$ messages holding this value, it outputs its input value, otherwise it defaults to 0. As $f < n/3$, by the same reasoning as in the lecture, it is not possible that two different correct nodes with different inputs keep their input value.

b) The issue is that o is not known to nodes that did not receive $n - f$ messages with value o. While it is straightforward to use binary consensus to agree on whether to output o or 0, in case the output is o, all correct nodes must be able to determine o.

c+d) Nodes that kept their original input in the algorithm from a) broadcast o (their input) a second time. Only if they receive $n-f$ messages with value o again, they use input 1 (meaning that they support choosing o) for the binary consensus instance. In all other cases, nodes will use input 0 (supporting the default value 0 as overall output). If a node receives $f+1$ or more messages with value o, they store o, returning it as output in case the binary consensus instance returns 1. Otherwise, the output 0.

To see that this works, observe that if any correct node uses input 1 for the binary consensus instance, this implies that all nodes receive $n - 2f > f$ times value o in the second round of broadcast, while no other value can appear more than f times (as correct nodes may only broadcast o). Thus, by validity of the binary consensus instance, all nodes know o and will output it in case the binary consensus instance returns 1 or all correct nodes use input 0 for the binary consensus instance and ultimately output 0. Together with the agreement property of the binary consensus instance, this implies that the described algorithm can be executed correctly and satisfies agreement. Validity is straightforward to verify by checking that the $n - f$ thresholds will be reached in both broadcasts and applying validity of the binary consensus instance.

The overhead is two rounds plus $\mathcal{O}(n^2 \log O)$ bits: two broadcasts of each node, encoding one of O possible values.

Task 2: As Fast as it Gets

Consider the task of binary consensus with up to $f < n$ crash faults.

a) Describe and prove correct an $(f + 1)$-round algorithm using 1-bit broadcasts for communication.

Solution

a) Each node $i \in V$ maintains an opinion op_i, initialized to its input value b_i. In each round, i broadcasts op_i, and sets it to 0 in case it receives a 0. Finally, it outputs op_i at the end of round $f + 1$.

Validity is trivial, as no node ever changes its opinion, if all opinions are identical. To show agreement, it is thus sufficient to prove that there is a round such that all opinions (of non-crashed nodes) are the same at the end of the round. As the algorithm runs for $f + 1$ rounds, there must be a round in which no node crashes. In this round, all nodes receive the same messages and set their opinions to the same value. More specifically, if there is any (non-crashed) node with opinion 0 at the beginning of the round, all non-crashed nodes receive a 0-message and set their opinion to 0. Otherwise, all non-crashed nodes have opinion 1 at the beginning and end of the round. As the running time and message size bounds are trivially satisfied, all desired properties hold.

Task 3*: Timing Issues

In this task, the goal is to transfer synchronous algorithms to the bounded-delay model with faults. Therefore, the setting is the same as in Chapter 4 of the lecture.

a) Use the Srikanth-Toueg algorithm to *simulate* synchronous execution of some given R-round synchronous algorithm in $\mathcal{O}(Rd)$ time, assuming that the execution is triggered by events at the individual nodes that are at most $\mathcal{O}(Rd)$ time apart (for a known bound).

b) Can you formalize what the term "simulates" here means precisely?

c) Things get a bit messy with randomization, as there typically are additional model assumptions needed for randomization to be useful. Figure out what these might be and whether this poses a problem!

d) Agree on your findings in the TA session!