Exercise Session 7
NBAC, TRB

Problem 1

Devise two algorithms that, without consensus, implement weaker specifications of NBAC *by replacing the termination property with the following ones:*

1. *Weak termination: Let p be a distinguished process, known to all other processes. If p does not crash then all correct processes eventually decide. Your algorithm may use a perfect failure detector.*
2. *Very weak termination: If no process crashes, then all processes decide. Is a failure detector needed to implement this algorithm?*

Solution

The first algorithm may rely on the globally known process p to enforce termination. The algorithm uses a perfect failure detector $\mathcal{P}$ and works as follows. All processes send their proposal over a point-to-point link to p. This process collects the proposals from all processes that $\mathcal{P}$ does not detect to have crashed. Once process p knows something from every process in the system, it may decide unilaterally. In particular, it decides COMMIT if all processes propose COMMIT and no process is detected by $\mathcal{P}$, and it decides ABORT otherwise, i.e., if some process proposes ABORT or is detected by $\mathcal{P}$ to have crashed. Process p then uses best-effort broadcast to send its decision to all processes. Any process that delivers the message with the decision from p decides accordingly. If p crashes, then all processes are blocked.

Of course, the algorithm could be improved in some cases, because the processes might figure out the decision by themselves, such as when p crashes after some correct process has decided, or when some correct process decides ABORT. However, the improvement does not always work: if all correct processes propose COMMIT but p crashes before any other process, then no correct process can decide. This algorithm is also known as the Two-Phase Commit (2PC) algorithm. It implements a variant of atomic commitment that is blocking.

The second algorithm is simpler because it only needs to satisfy termination if all processes are correct. All processes use best-effort broadcast to send their proposals to all processes. Every process waits to deliver proposals from all other processes. If a process obtains the proposal COMMIT from all processes, then it decides COMMIT; otherwise, it decides ABORT. Note that this algorithm does not make use of any failure detector.

Problem 2

Can we implement TRB with the eventually perfect failure detector $\diamond P$, if we assume that at least one process can crash?

Solution

The answer is no. Consider an instance *trb* of TRB with sender process s. We show that it is impossible to implement TRB from an eventually perfect failure-detector primitive $\diamond P$, if even one process can crash.

Consider an execution E_1 , in which process s crashes initially and observe the possible actions for some correct process p: due to the termination property of TRB, there must be a time T at which p trb-delivers $\perp$.

Consider a second execution E_2 that is similar to E_1 up to time T, except that the sender s is correct and *trb-broadcasts* some message m, but all communication messages to and from s are delayed until after time T. The failure detector behaves in E_2 as in E_1 until after time T. This is possible because the failure detector is only eventually perfect. Up to time T, process p cannot distinguish E_1 from E_2 and *trb-delivers* $\perp$. According to the *agreement* property of TRB, process s must *trb-deliver* as well, and s delivers exactly one message due to the *termination* property. But this contradicts the *validity* property of TRB, since s is correct, has *trb-broadcast* some message $m \neq \perp$, and must *trb-deliver* m.