Exercise Session 6
Non-Blocking Atomic Commit

November 12, 2012

Problem 1

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

- *weak termination: let p_i be some process; if p_i does not crash then all correct processes eventually decide;*
- *very weak termination: if no process crashes, then all processes decide.*

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 NBAC *with the eventually perfect failure detector $\diamond P$ if we assume that at least one process can crash? What if we consider a weaker specification of* NBAC *where the agreement property is not required?*

The answer is no. To explain why, we consider an execution E_1 , where all processes are correct and propose COMMIT, except for some process p that proposes ABORT and crashes initially, without sending any message. All correct processes must therefore decide ABORT in E_1, as deciding COMMIT would violate the commit-validity property. Let T be the time at which the first (correct) process q decides ABORT. It does so presumably after receiving some output of $\diamond\mathcal{P}$, which indicated that p crashed.

Consider now an execution E_2 that is similar to E_1 except that p is correct and proposes COMMIT, but all its messages are delayed until after time T. The failure detector behaves in E_2 as in E_1 until time T

and suspects p to have crashed; this is possible because $\diamond\mathcal{P}$ is only eventually perfect. Hence, no process apart from p can distinguish between E_1 and E_2 and q also decides ABORT in E_2 . But this violates the abort-validity property, as all processes are correct and propose COMMIT, yet they decide ABORT.

In this argument, the (uniform or regular) agreement property of NBAC was not explicitly needed. This shows that even a specification of NBAC where agreement was not needed could not be implemented with an eventually perfect failure detector if some process crashes.