6. Comparing the Atomic Commitment and Consensus Problems

Bernadette Charron-Bost
LIX, École polytechnique 91128 Palaiseau Cedex, France
charron@lix.polytechnique.fr

6.1 Two Agreement Problems: Consensus and NB-AC

Reaching agreement in a distributed system is a fundamental issue of both theoretical and practical importance. Consensus and non-blocking atomic commitment are two well-known versions of this paradigm. The Consensus problem considers a fixed collection of processors each of which has an initial value drawn from some domain $V$, and processors must eventually decide on the same value $1$; moreover, the decision value must be the initial value of some processor. The non-blocking atomic commitment (NB-AC) problem arises in distributed database systems to ensure the consistent termination of transactions. Each process that participates in the processing of a database transaction arrives at an initial opinion (vote) about whether to commit the transaction or abort it. Processes must eventually reach a common decision (commit or abort). The decision to commit may be reached only if all processes initially vote to commit. In this case, “commit” must be reached if there is no failure.

As observed by Hadzilacos in [6.16], the binary Consensus and NB-AC specifications are very similar. They only differ in their validity conditions, i.e., the conditions describing the decision values that are permitted: in the Consensus problem, the possible decision values are related to the initial values only, contrary to the NB-AC problem for which the decision values depend on both input values and failure pattern. A closer look at these validity conditions shows that they cannot be compared. So in the strict standpoint of specification, Consensus and NB-AC are incomparable. But what about their solvability and their complexity? In other words, is one problem harder to solve than the other?

This note is devoted to various results on this question I have recently obtained, most of them in collaboration, for the crash failure model. Firstly, I recall the ones in my joint work with F. Le Fessant [6.6] concerning synchronous systems: we have proved that in this setting, Consensus and NB-AC are two very similar problems which are both solvable whatever the environment, and with the same time complexity. Then I give some impossibility results for the NB-AC problem in non-synchronous systems in which Consensus is solvable: in such systems, NB-AC is thereby a harder problem than Consensus. Finally, I present two theorems due to S. Toueg and myself which show that, in asynchronous systems, Consensus and NB-AC are incomparable in almost all environments. I complete those theorems by examining what information about failures is necessary and sufficient to solve NB-AC in asynchronous systems.

1 More exactly, the problem that is considered here is uniform Consensus: no two (correct or faulty) processes can decide differently.

© Springer-Verlag Berlin Heidelberg 2003
6.2 When NB-AC and Consensus Are Similar

In synchronous systems, Consensus and NB-AC are both solvable no matter how many processes fail, and so are equivalent in terms of solvability. Actually they are equivalent in a stronger sense: in the presence of up to \( t \) failures, \( t + 1 \) rounds\(^2\) are necessary and sufficient in the worst case execution to solve Consensus as well as NB-AC [6.19]. Following [6.9], one refines this comparison by discriminating runs according to the number of failures \( f \) that actually occur (\( 0 \leq f \leq t \)). Charron-Bost and Schiper [6.7] proved that both Consensus and NB-AC require at least \( f + 2 \) rounds if \( f < t - 1 \), and only \( f + 1 \) rounds if \( t - 1 \leq f \leq t \) (subsequently, Keidar and Rajsbaum [6.17] have given another proof of this lower bound).

One may wonder whether these lower bounds for early deciding Consensus and NB-AC algorithms are tight. A Consensus algorithm is presented in [6.7] that achieves these lower bounds. Unfortunately, this algorithm cannot be easily adapted for the NB-AC problem. As noticed by Lamport in [6.18], “a [popular] belief is that a three-phase commit protocol à la Skeen [6.20] is needed” for solving NB-AC. It is not exactly clear what that means, but it seems to imply that at least three rounds are required in failure-free runs for deciding. The \( f + 2 \) lower bound would be therefore weak in the case of NB-AC. In [6.6], F. Le Fessant and I show that this intuition is incorrect: we devise a general algorithm for both Consensus and NB-AC which achieves the general lower bounds for early deciding established in [6.7]. This proves that even when considering the question of early deciding, Consensus and NB-AC remain two equivalent problems.

6.3 When NB-AC Is Harder than Consensus

As observed by Gray [6.12], the impossibility of NB-AC in an asynchronous system can be established by quite a simple argument even if the maximum number of failures is one, and so does not require the subtle proof given in [6.11] for Consensus. The argument is based on the following two points:

1. In an asynchronous system, a process may be so slow that it executes its first step after the other processes have decided; the decision value is the same as if the slow process had initially crashed.
2. The decision value of any run with one initial crash is necessarily “abort”.

This argument yields the following three impossibility results in systems subject to a single crash failure:

1. Even with initial failure, NB-AC cannot be solved in an asynchronous system.
2. There is no asynchronous randomized algorithm solving NB-AC.
3. NB-AC cannot be solved in any of the partially synchronous models described in [6.10]\(^3\).

\(^2\) In a synchronous algorithm, a run proceeds in \textit{rounds}: in each round, every process sends messages to all processes, receives all the messages sent to it in the round, and then changes its state. Time complexity is then measured in terms of the number of rounds necessary to produce all the required outputs.

\(^3\) Using the same simple argument, Guerraoui [6.13] showed this result in some partially synchronous systems defined in terms of unreliable failure detectors [6.3]. Obviously, the argument applies to the more general and realistic partially synchronous models of [6.10].