Impossibility Results in the Presence of Multiple Faulty Processes
(Preliminary Version)

Gadi Taubenfeld*  Shumel Katz†  Shlomo Moran‡

Abstract. We investigate the impossibility of solving certain problems in an unreliable distributed system where multiple processes may fail. We assume undetectable crash failures which means that a process may become faulty at any time during an execution and that no event can happen on a process after it fails. A sufficient condition is provided for the unsolvability of problems in the presence of multiple faulty processes. Several problems are shown to be solvable in the presence of $t - 1$ faulty processes but not in the presence of $t$ faulty processes for any $t$. These problems are variants of problems which are unsolvable in the presence of a single faulty process (such as consensus, choosing a leader, ranking, matching). In order to prove the impossibility result a contradiction is shown among a set of axioms which characterize any fault-tolerant protocol solving the problems we treat. In the course of the proof, we present two results that appear to be of independent interest: first, we show that for any protocol there is a computation in which some process is a splitter. This process can split the possible outputs of the protocol to two disjoint sets. In case that the protocol is also fault-tolerant, then this splitter must be a decider, that can split its own output values into two different singletons. These results generalize and expand known results for asynchronous systems.

1 Introduction

In this paper we investigate the possibility and impossibility of solving certain problems in an unreliable distributed system where a number of processes may fail. We assume undetectable crash failures which means that no event can happen on a process after it fails and that failures are undetectable. For any $1 \leq t < n$, where $n$ is the number of processes, we define a class of problems which cannot be solved in a completely asynchronous system where $t$ processes may fail. This implies a (necessary) condition for solving a problem in such an unreliable system. These results generalize previously known impossibility results for completely asynchronous systems, and prove new results.

*Computer Science Department, Yale University, New Haven, CT 06520. Supported in part by the National Science Foundation under grant DCR-8405478, by the Hebrew Technical Institute scholarship, and by the Gutwirth Fellowship.

†Computer Science Department, Technion, Haifa 32000, Israel.

‡Computer Science Department, Technion, Haifa 32000, Israel. Supported in part by Technion V.P.R. Funds - Wellner Research Fund, and by the Foundation for Research in Electronics, Computers and Communications, administrated by the Israel Academy of Sciences and Humanities.
Various authors have investigated the nature of systems where only a single process may fail (i.e., $t = 1$). It is proven in [FLP] that in asynchronous systems there cannot exist a nontrivial consensus protocol that tolerates even a single process (crash) failure. This fundamental result has been extended to other models of computation [DDS,DLS]. Various extensions [MW, Ta, BMZ], also for a single fault, prove the impossibility of other problems using several new techniques. Other recent works point out some specific problems that can be solved in asynchronous systems with numerous faulty processes, and prove impossibility results for other problems [ABDKPR,BW,DDS]. In [TKM2] a necessary and sufficient condition is provided for solving problems in an unreliable asynchronous message passing systems where undetectable initial failures may occur. Initial failures are a very weak type of failures where it is assumed that processes may fail only prior to the execution. Results for asynchronous shared memory systems which support only atomic read and write operations, similar to those presented here appear in [TM].

Define an input vector to be a vector $\bar{a} = (a_1, ..., a_n)$, where $a_i$ is the input value of process $P_i$. A crucial assumption in all the above results (for a single process failure) is that the set of input vectors is "large enough". To demonstrate this fact, consider the consensus problem where only two input vectors are possible: either all processes read as input the value "zero" or all processes read as input the value "one". It is easy to see that under this restriction, the problem can be solved assuming any number of process failures. One of the consequences of our result is to identify a property (or a promise [ESY]) which a set of input vectors should satisfy so that a problem can be solved in the presence of $t - 1$ faulty processes but not in the presence of $t$ faulty processes.

We show variants of the problems which are known to be unsolvable in the presence of a single faulty process (such as consensus, choosing a leader, ranking, matching, and sorting) and prove that the variants can be solved in the presence of $t - 1$ faulty processes but not in the presence of $t$ faulty processes for any $t$. An example is the consensus problem where the promise is that for each input vector, $|\#1 - \#0| \geq t$. (i.e., the absolute difference between the number of ones and the number of zeroes is at least $t$.)

The proof of our result is constructed as follows. We first identify a class of protocols that cannot tolerate the failure of $t$ processes, when operating in a completely asynchronous system. Then, we identify those problems which force every protocol which solves them to belong to the above class of protocols. Hence, these problems cannot be solved in a completely asynchronous system where $t$ processes may fail.

As in [FLP], we differentiate between a process having reached a decision, and a stage at which the eventual decision to be reached by a process is uniquely determined (but usually not yet known at a process). The class of protocols for which we prove the impossibility result is characterized by two requirements on the possible input and decision (output) values of each member in the class. For the input, it is required that (for each protocol) there exists a group of at least $n - t$ processes and there exist input values such that after all the $n - t$ processes in the group read these input values, the eventual decision value of at least one of them is still not uniquely determined. As for the decision values, the decision of different processes should have the following mutual dependency: the eventual decision value of any (single) process is uniquely determined as soon as all other processes decide.

In order to prove the above result for protocols, we use an axiomatic approach for proving properties of protocols (and problems) due to Chandy and Misra [CM1,CM2]. The idea is to capture the main features of the model and the features of the class of