Yale University
Department of Computer Science

Leader Election in the Presence of $n-1$ Initial Failures

Gadi Taubenfeld

YALEU/DCS/TR-709
May 1989

This work was supported in part by the National Science Foundation under grant CCR-8405478, and by the Hebrew Technical Institute scholarship.

Leader Election in the Presence of $n - 1$ Initial Failures

Gadi Taubenfeld*
Computer Science Department, Yale University, New Haven, CT 06520

Key words: leader election, shared memory, crash failures, initial failures.

1 Introduction

We present a deterministic leader election protocol that can tolerate up to $n-1$ undetectable initial failure, where n is the number of processes. We assume an asynchronous shared memory environment which support only atomic read and write operations.

Initial failures are a very weak type of failures where it is assumed that processes may fail only prior to the execution and that no event can happen on a process after it fails. That is, once a process starts operating it is guaranteed that it will never fail. Initial failures are a special case of crash (fail-stop) failures in which a process may become faulty at any time during an execution.

In an election protocol each processes reads an input value from some well order set, and exactly one process is to decide on a distinguished value from an arbitrary set of (output) values. An election protocol can tolerate up to $n - 1$ initial failures if in spite of a failure of any group of up to $n - 1$ processes at the beginning of the computation, each of the remaining processes eventually terminates, and exactly one process is elected as a leader.

Initial failures may occur in situations such as recovery from a breakdown of a system. Several protocols were designed to properly operate in a message passing model where initial failures may occur. A protocol that solves the consensus problem which can tolerate initial failures of up to (not including) half of the processes is presented in [FLP]. Protocols for leader election and spanning tree construction which can also tolerate initial failures of up to half of the processes are designed in [BKWZ].

The results in [Abr,CIL,Her,LA] proves the nonexistence of a (nontrivial) deterministic consensus (and election) protocol that can tolerate $n-1$ crash failure, for a shared memory model. Randomized consensus and election protocols that can tolerate $n - 1$ crash failures are presented in [Abr,CIL].

My interest in designing the election protocol arose in a study by Shlomo Moran and myself, about the solvability of problems in an unreliable shared memory model. In [MT, Section 7], a necessary and sufficient condition is provided for solving problems in a shared memory model in the presence of multiple initial failures. The correctness proof of that condition is based on the existence of the leader election protocol presented in the next section.

*Supported in part by the National Science Foundation under grant CCR-8405478, and by the Hebrew Technical Institute scholarship.

```
THE PROTOCOL FOR PROCESS $i$, $(1 \leq i \leq n)$ :
All registers are initialized to zero.

$decide_i : 0..1$;
$wake_i : 0..n$;   % used for counting the number of processes that wake up (i.e., are correct)
$faulty_i : 0..n$; % the number of processes that did not wake up according to process $i$
                   % counting, plus one process

$wake_i := 1$;     % process $i$ signals to the other processes that it woke up and counts itself
$decide_i := 1$;

for $j = 1$ to $n$ do          % counting the number of processes that wake up
  if $(wake_j \neq 0)$ and $(i \neq j)$ then $wake_i := wake_i + 1$ end-if
end-do ;
$faulty_i := n - wake_i + 1$; % the number of processes that may be faulty, plus one process
                              % We add one to ensure that $faulty_i \neq 0$ at this point

for $j = 1$ to $n$ do
  if $wake_j \neq 0$ then       % process $j$ is correct
    while $faulty_j = 0$ do skip end-while                          % busy waiting
    if $(faulty_j, j) > (faulty_i, i)$ then $decide_i := 0$ end-if   % lexicographic order
  end-if
end-do;

if $decide_i = 0$ then "I am not the leader"
              else "I am the leader"
end-if.
```

Figure 1: A $(n-1)$-resilent leader election protocol.

2 The protocol and its verification

A protocol that solves the election problem and which can tolerate $n-1$ initial failures is presented in Figure 1. We assume that all the registers are initialized to zero. The general idea behind the protocol is the following. Each process when it wakes up, tries to count (only once) the number of processes that woke up so far. Since, processes wake up at different times, the counting of different processes may differ. The process that is eventually elected is the one with the biggest ID among those processes that have counted the smallest number of woke up processes. We now give a formal correctness proof of the protocol. In the sequel, we say that process i *woke up* if $wake_i \neq 0$, process i *finished counting* if $faulty_i \neq 0$, and processes i is *elected* if the process has terminated and $decide_i = 1$.

Theorem 1: *Every processes that wakes up eventually terminates.*

Proof: Since we assume initial failures, a process that wakes up can not fail. Thus, any process that wakes up eventually finish counting. What we have to show is that a process can not loop forever in the while loop. However a process that enters the while loop, waits for some other process which already woke up to finish counting. Since any process that wakes up eventually finish counting, no process busy-waits forever. □

Theorem 2: *Exactly one process is elected.*

The proof of Theorem 2, follows immediately from the next two lemmas.

Observation 1: *If process i finished counting and process j has not waken up yet, then process j can not be elected.*

Proof: If process i finished counting and process j has not waken up yet, then at any later time $faulty_i > faulty_j$. Hence, when process j will find out that $(faulty_i, i) > (faulty_j, j)$, it will set its decision bit to zero, and will not be elected. □

Lemma 1: *At most one process is elected.*

Proof: Assume to the contrary that both process i and process j $(i \neq j)$ are elected. By Observation 1, when process i finished counting, process j already woke up, and vice versa. Hence, process i and process j have to wait until both of them finish counting, after which the pairs $(faulty_i, i)$ and $(faulty_j, j)$ have to be compared by each one of them before it can be elected. Since $(faulty_i, i) \neq (faulty_j, j)$, either process i or process j has to set its decision bit to zero, and it can not be elected. A contradiction. □

Lemma 2: *At least one process is elected.*

Proof: For each process j, if it has terminated without being elected then there must be some other process that woke up, say i, such that $(faulty_i, i) > (faulty_j, j)$. Since every process that wakes up eventually terminates, at least for one process that terminates, say k, there does not exist any process, say i, such that $(faulty_i, i) > (faulty_k, k)$. Hence, process k must be elected. □

Acknowledgements: I am grateful to Shlomo Moran for helpful discussions concerning this work.

References

[Abr] Abrahamson, K. On achieving consensus using shared memory, *ACM-PODC* 1988, 291-302.

[BKWZ] Bar-Yehuda, R., Kutten, S., Wolfstahl, Y., and Zaks, S. Making distributed spanning tree algorithms fault-resilient, *STACS* 87, LNCS no. 247, 432-445.

[CIL] Chor, B., Israeli, A., and Li, M. On processor coordination using asynchronous hardware, *ACM-PODC* 1987, 86-97.

[FLP] Fischer, M., Lynch, N., Paterson, M. Impossibility of distributed consensus with one faulty process, *JACM* Vol. 32, No. 2, 1985, 374-382.

[Her] Herlihy, P.M. Impossibility and universality results for wait-free synchronization, *ACM-PODC* 1988, 276-290.

[LA] Loui, C.M., and Abu-Amara, H. Memory requirements for agreement among unreliable asynchronous processes, *Advances in Computing Research*, Vol. 4, 1988, 163-183.

[MT] Moran, S., and Taubenfeld, G. Impossibility results in a shared memory environment, *YALEU/DCS/TR-708*, 1989.