A Simple Bivalency Proof that t-Resilient Consensus Requires $t+1$ Rounds*

Marcos Kawazoe Aguilera and Sam Toueg

aguilera@cs.cornell.edu sam@cs.cornell.edu

Department of Computer Science
Upson Hall, Cornell University
Ithaca, NY 14853-7501, USA.

Keywords: distributed computing, fault-tolerance, consensus, lower bound, synchronous system, bivalency

January 28, 2000

Abstract

We use a straightforward bivalency argument borrowed from Fischer et al. [7] to show that in a synchronous system with up to t crash failures solving consensus requires at least $t+1$ rounds. The proof is simpler and more intuitive than the traditional one: It uses an easy forward induction rather than a more complex backward induction which needs the induction hypothesis several times.

1 Background

A fundamental result of distributed computing is that solving consensus in a synchronous system with up to t process crashes requires at least $t+1$ rounds [9, 8, 5]. The traditional proof of this result proceeds by a rather complex backward induction that uses the induction hypothesis several times [10]. In this note, we provide a much simpler and intuitive proof: it uses an easy forward induction and it is based on a standard bivalency argument. Proofs similar to ours have been independently found by Moses and Rajsbaum [11], and by Bar-Joseph and Ben-Or [2] (see Section 3 for details on related work).

In the following, we consider systems where processes proceed in synchronized rounds: in each round, every process sends messages to other processes, receives all the messages sent to it in that round, and changes state accordingly. When a process crashes in a round, it sends a subset of the messages that it intends to send in that round, and does not execute any subsequent rounds. A *correct* process is one that never crashes.

In the consensus problem, every process starts with some *initial value* and must make an irrevocable *decision* on a value such that:

Agreement: No two correct processes decide differently.

Validity: If some correct process decides v, then v is the initial value of some process.

Termination: Every correct process must eventually decide some value.

*Research partially supported by the NSF grant CCR-9711403 and by an Olin Fellowship.

2 The Proof

We now show that any consensus algorithm that tolerates t crashes requires $t+1$ rounds. Roughly speaking, the proof proceeds by contradiction as follows. Suppose there is a consensus algorithm A that tolerates up to t crashes and always terminates in t rounds. We first show that in any run of A, the configuration at the beginning of round t must be univalent. We then obtain a contradiction by constructing a run of A that is bivalent at the beginning of round t. This run is obtained by starting from a bivalent initial configuration and extending it one round at a time, while maintaining bivalency. Each one-round extension may require the killing of a process.

Theorem 1 *Consider a synchronous round-based system $\mathcal{S}$ with n processes and at most t crash failures such that at most one process crashes in each round. If $n > t + 1$ then there is no algorithm that solves consensus in t rounds in $\mathcal{S}$.*

The proof is by contradiction. Suppose there is an algorithm A that solves consensus in t rounds in $\mathcal{S}$. Without loss of generality, we can assume that A is loquacious, i.e., at every round, each process is supposed to send a message to every process.

We consider the configuration of the system $\mathcal{S}$ at the end of each round (this is also the configuration of the system just before the start of the next round). Such a configuration is just the state of each process (which also indicates the current round number and whether it has crashed in a previous round). Informally, a configuration C *is* 0*-valent* [1*-valent*] if starting from C the only possible decision value of correct processes is 0 [1]; C *is univalent* it is either 0-valent or 1-valent; C *is bivalent* if it is not univalent.

In the following, a *k-round partial run r_k* denotes an execution of algorithm A up to the end of round k. Consider the configuration C_k at the end of round k of partial run r_k. We say that r_k *is* 0*-valent*, 1*-valent*, *univalent*, or *bivalent* if C_k is 0-valent, 1-valent, univalent, or bivalent, respectively.

We proceed by proving three lemmata. The third one contradicts the first and thus completes the proof of the theorem.

Lemma 1 *Any $(t-1)$-round partial run r_{t-1} is univalent.*

Proof: The proof is by contradiction. Suppose there is a bivalent $(t-1)$-round partial run r_{t-1}. Let r^0 be the t-round run obtained by extending r_{t-1} by one round such that no process crashes in round t. Without loss of generality assume that all correct processes decide 0 in r^0. Since partial run r_{t-1} is bivalent, there is at least one t-round run r^1 that extends r_{t-1} such that all correct processes decide 1. Note that in round t of r^1: (a) exactly one process p must crash (recall that in each run at most one process crashes per round), and (b) p must fail to send a message to at least one correct process, say c.

Construct run $r^{0,1}$ which is identical to r^1, except that p sends its message to c. Let c' be a process that does not crash in $r^{0,1}$ and is different from c. Such a process must exist since $n > t + 1$ implies that there are at least two correct processes in the system. Note that: (a) c cannot distinguish between $r^{0,1}$ and r^0; (b) c' cannot distinguish between $r^{0,1}$ and r^1. By (a), c decides 0 in $r^{0,1}$, while by (b) c' decides 1 in $r^{0,1}$ — a violation of the agreement property of consensus. □

Lemma 2 *There is a bivalent initial configuration.*

Proof: (Same as in Fischer et al. [7]) Suppose, for contradiction, that every initial configuration is univalent. Consider the initial configurations C^0 and C^1 such that all processes have initial value 0 and 1, respectively. By the validity property of consensus, C^0 is 0-valent and C^1 is 1-valent. Clearly, there are two initial configurations that differ by the initial value of only one process p, such that one is 0-valent and the other is 1-valent. We can easily reach a contradiction by crashing p at the beginning of round 1 (before it sends any messages to any process). □

Lemma 3 *There is a bivalent* $(t-1)$*-round partial run* r_{t-1}.

Proof: We show by induction on k that for each k, $0 \le k \le t-1$, there is a bivalent k-round partial run r_k.
BASIS: By Lemma 2, there is some bivalent initial configuration C_0. For $k = 0$, let r_0 be the 0-round partial run that ends in C_0.
INDUCTION STEP: Suppose $0 \le k < t-1$. Let r_k be a bivalent k-round partial run. We now show that r_k can be extended by one round into a bivalent $(k+1)$-round partial run r_{k+1}. Assume, for contradiction, that every one-round extension of r_k is univalent.

Let r^*_{k+1} be the partial run obtained by extending r_k by one round such that no new crashes occur. Partial run r^*_{k+1} is univalent. Without loss of generality assume it is 1-valent. Since r_k is bivalent, and every one-round extension of r_k is univalent, there is at least one one-round extension r^0_{k+1} of r_k that is 0-valent.

Note that r^*_{k+1} and r^0_{k+1} must differ in round $k+1$ (and only in that round). Since round $k+1$ of r^*_{k+1} is failure-free, there must be exactly one process p that crashes in round $k+1$ of r^0_{k+1} (recall that in each run, at most one process crashes per round). Since p crashes in round $k+1$ of r^0_{k+1} it may fail to send a message to some processes, say to $q_1, q_2, \ldots, q_m$, where $0 \le m \le n$.[1]

Starting from r^0_{k+1}, we now define $(k+1)$-round partial runs $r^1_{k+1}, \ldots, r^m_{k+1}$ as follows. For every j, $1 \le j \le m$, r^j_{k+1} is identical to r^{j-1}_{k+1} except that p sends a message to q_j before it crashes in round $k+1$. Note that for every j, $0 \le j \le m$, r^j_{k+1} is univalent. There are two possible cases:

1. For all j, $0 \le j \le m$, r^j_{k+1} is 0-valent. So r^m_{k+1} and r^*_{k+1} are 0-valent and 1-valent, respectively. The only difference between r^m_{k+1} and r^*_{k+1} is that p crashes at the end of round $k+1$ in r^m_{k+1}, while p is correct up to and including round $k+1$ in r^*_{k+1}. Consider the following run r extending r^*_{k+1}. Process p crashes at the beginning of round $k+2$ (before it sends any messages in that round), and there are no more crashes. Since r^*_{k+1} is 1-valent, all correct processes decide 1 in run r. For every process except p, run r is indistinguishable from the run r' that extends r^m_{k+1} such that no process crashes after round $k+1$. But all correct processes decide 0 in r' (because r^m_{k+1} is 0-valent) — a contradiction.

2. There is a j, $1 \le j \le m$, such that r^{j-1}_{k+1} is 0-valent while r^j_{k+1} is 1-valent. Extend partial runs r^{j-1}_{k+1} and r^j_{k+1} into runs r and r', respectively, by crashing process q_j at the beginning of round $k+2$ (before it sends any message in that round),[2] and continuing with no additional crashes. Note that (a) no process except q_j can distinguish between r and r', and (b) all correct processes must decide 0 in r and 1 in r' — a contradiction.

□

3 Related Work

The $(t+1)$-rounds lower bound for consensus was first proved for synchronous systems with Byzantine process failures [6]. This lower bound was extended first to systems with Byzantine failures and message authentication [3, 4], and then to systems with crash failures [9, 8]. A refinement of the $(t+1)$-rounds lower bound was later obtained by determining the number of rounds necessary to reach (simultaneous) consensus, given the pattern in which crashes occur [5]. The lower-bound proof in [6] is for byzantine failures, and it is not clear that it can be extended to more benign models of failures. On the other hand, the lower-bound proofs in [3, 4, 9, 8] are based on a relatively complex backward induction, and they do not use bivalency arguments.

[1] It is possible that in round $k+1$ of r^0_{k+1} process p sends a message to *every* process, and then crashes at the end of this round. In this case, $m = 0$.

[2] If q_j already crashed before round $k+2$, we don't crash it in round $k+2$.

Moses and Rajsbaum [11], and Bar-Joseph and Ben-Or [2] have independently found a $(t+1)$-rounds lower bound proof that is similar to ours. In [11], Moses and Rajsbaum introduce the notion of *layering*, and use it to provide a unified analysis of consensus that can be applied to several models of distributed computation, including asynchronous shared memory with crash failures, asynchronous and synchronous message passing with crash failures, and synchronous message passing with mobile failures.[3] A layering is defined as a function S that maps a state to a set of (not necessarily immediate) successor states; intuitively, it allows one to focus on "interesting states" within "interesting runs". Using layering, [11] proves some general results on consensus. In particular, a lemma shows that given a bivalent state x, if $S(x)$ satisfies a certain condition then x has a successor state that is also bivalent. For the round-based synchronous model, [11] defines a specific layering S,[4] and uses this lemma to obtain a run with a bivalent state at the end of $t-1$ rounds. To complete the $(t+1)$-rounds lower bound proof, [11] shows that after reaching a bivalent state, two extra rounds are necessary for the decision.

In [2], Bar-Joseph and Ben-Or show tight upper and lower bounds of $\Theta(t/\sqrt{n \log n})$ on the expected number of rounds needed for randomized consensus in synchronous systems with crash failures, and with a full-information and adaptive adversary. While [2] does not explicitly show the $(t+1)$-round lower bound result, it contains the essence of the bivalency-based proof of our paper.

The bivalency argument was first introduced by Fischer, Lynch and Paterson [7] in the context of *asynchronous* systems (to prove that consensus cannot be solved in the presence of crashes). To the best of our knowledge, the first paper to use a bivalency argument in the context of *synchronous* systems was [12]. Specifically, bivalency is used in [12] to show a general result on the *impossibility* of consensus that can be applied to synchronous systems where one process can fail per round, but there is no bound on the number of rounds with failures (e.g., a system with recurrent mobile failures).

Acknowledgements

We thank Bernadette Charron-Bost for her comments on an earlier draft, and the anonymous referees for useful suggestions.

References

[1] *Proceedings of theSeventeenth ACM Symposium on Principles of Distributed Computing*, June 1998.

[2] Ziv Bar-Joseph and Michael Ben-Or. A tight lower bound for randomized synchronous consensus. In *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing* [1], pages 193–199.

[3] Richard A. DeMillo, Nancy A. Lynch, and Michael J. Merritt. Cryptographic protocols. In *Proceedings of the Fourteenth ACM Symposium on Theory of Computing*, pages 383–400, May 1982.

[4] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine Agreement. *SIAM Journal on Computing*, 12(4):656–666, November 1983.

[5] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, October 1990.

[3]In this latter model, in each round one process may fail to send some of its messages. The failing process can change from round to round, and hence the term "mobile".

[4]Roughly speaking, if x is the state at the beginning of a round, then $S(x)$ is the set of states that can be reached by either a failure-free round, or, if there were fewer than t failures so far, by a round in which some process crashes and fails to send a message to a prefix of the processes.

[6] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14:183–186, June 1982.

[7] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[8] Vassos Hadzilacos. A lower bound for byzantine agreement with fail-stop processors. Technical Report 21-83, Department of Computer Science, Harvard University, July 1983.

[9] Leslie Lamport and Michael Fischer. Byzantine Generals and Transaction Commit protocols. Technical Report 62, Comp. Sci. Lab., SRI International, April 1982.

[10] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[11] Yoram Moses and Sergio Rajsbaum. The unified structure of consensus: a layered analysis approach. In *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing* [1], pages 123–132.

[12] Nicola Santoro and Peter Widmayer. Time is not a healer. In B. Monien and R. Cori, editors, *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, volume 349 of *LNCS*, pages 304–313, Berlin, February 1989. Springer.