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, synchronous system, bivalency

September 1, 1998

Abstract

We use a straightforward bivalency argument borrowed from [2] 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. The traditional proof of this result proceeds by a rather complex backward induction that uses the induction hypothesis several times [3]. In this note, we provide a much simpler proof based on a standard bivalency argument.

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 NSF grants CCR-9402896 and CCR-9711403, by ARPA/ONR grant N00014-96-1-1014, 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 [2].) 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, $0 \leq k \leq t-1$, 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 $0 \leq k < t-1$. 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 $0 \leq m \leq n$.[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, $1 \leq j \leq m$, 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, $0 \leq j \leq m$, r^j_{k+1} is univalent. There are two possible cases:

1. For all j, $0 \leq j \leq m$, 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, $1 \leq j \leq m$, 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

[4] and [1] have independently come up with a proof that is similar to ours. The bivalency argument used in this note originally appeared in [2] to show a different result, namely that consensus cannot

[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 $k+2$.

be solved in asynchronous systems subject to failures. As far as we know, [5] were the first to use the bivalency argument of [2] in the context of synchronous systems. The traditional proof that t-resilient consensus requires $t + 1$ rounds is in [3].

Acknowledgements

We thank Bernadette Charron-Bost for her comments on an earlier draft.

References

[1] Ziv Bar-Joseph and Michael Ben-Or, August 1998. Private communication.

[2] 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.

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

[4] 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*, pages 123–132, June 1998.

[5] 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.