A Simple Proof of the Uniform Consensus Synchronous Lower Bound*

Idit Keidar† Sergio Rajsbaum‡

November 6, 2002

Abstract

We give a simple and intuitive proof of a $f+2$ round lower bound for uniform consensus. That is, we show that for every uniform consensus algorithm tolerating t failures, and for every $f \leq t-2$, there is an execution with f failures that requires $f+2$ rounds.
Keywords: distributed computing; fault tolerance; lower bounds; consensus.

1 Introduction

In a *consensus* problem, each process has an input in $\{0,1\}$ and can decide on an output in $\{0,1\}$[1], such that the following conditions are satisfied: (1) *termination*: each correct process decides on an output; (2) *agreement*: every two correct processes that decide decide on the same output; and (3) *validity*: the output is the input of one of the processes. The *uniform consensus* problem replaces agreement with *uniform agreement*: every two processes (correct or faulty) that decide, decide on the same output.

We consider a standard synchronous message-passing crash failure model [12] with n processes $p_1, \ldots, p_n$, of which at most t can crash, $t < n$. We assume a fully connected network. In a *round* of an algorithm each process sends messages to any subset of the processes, receives messages, and does local processing. If a process fails in a given round, then any subset of the messages it sends in this round can be lost, and the process does not participate in any later rounds.

In this paper we present a new lower bound proof for *early-deciding* [3] uniform consensus. We show that in executions with f failures, uniform consensus requires $f+2$ rounds, for $0 \leq f \leq t-2$. This is in contrast to non-uniform consensus, which can be solved in $f+1$ rounds [12]. Our proof is based on a very simple forward induction argument.

A standard technique for showing impossibility results and lower bounds related to consensus by forward induction is based on a notion of *bivalency*, that is, the existence of a state from which two different executions lead to different decisions. This technique was first introduced in [7] and was later used to prove numerous lower bound and impossibility results (see, e.g., [13, 1, 6]). Interestingly, the numerous bivalency proofs occurring in the literature do not make a strong enough use of validity to be able to prove the $f+2$ lower bound for uniform consensus of this paper, as

*To appear in Information Processing Letters (IPL), 85(1), pp. 47–52, December 2002. Preliminary version appeared in [10].

†MIT Lab for Computer Science and Technion Dept. of EE, current address: 545 Technology Square, NE43-367, Cambridge, MA 02139, U.S.A.

‡Compaq Cambridge Research Laboratory, One Cambridge Center, Cambridge, MA 02142-1612, U.S.A. On leave from Instituto de Matemáticas, UNAM.

[1]For our lower bound purposes, it is enough to consider only binary consensus.

we show in Section 3. Therefore, we base our proof on a stronger notion than bivalency, namely, the existence of *two* states that lead to different decision values in *failure-free* executions[2]. Using this new notion, we give, for the first time, a proof by forward induction of the uniform consensus lower bound.

The uniform consensus lower bound has implications for the consensus problem in certain partial synchrony models and asynchronous models with unreliable failure detectors: In such models, any algorithm for consensus also solves uniform consensus [8, 9, 10]. Therefore, our lower bound for the case of $f = 0$ implies a lower bound of two rounds for consensus in failure-free executions of such models (see [10]). This bound is tight: there are algorithms for consensus with unreliable failure detectors that decide in two rounds in all failure-free suspicion-free executions, e.g., [14].

In our proof, we use a weaker version of the validity condition defined as follows: for every value $v \in \{0, 1\}$ there is a failure-free execution in which some process decides v. By considering a weaker problem, we strengthen the result. In particular, this formulation of uniform consensus is also weaker than *non-blocking atomic commit* [15, 5], and therefore, the lower bound we prove in this paper also applies to non-blocking atomic commit.

Related work

The lower bound presented here was previously shown by Charron-Bost and Schiper [2]. However, their proof proceeds by a rather involved backward induction, (similar to the original proofs of the $f+1$ round lower bound for consensus [12]) and is therefore difficult to follow. Moreover, the proof in [2] does not cover the case of $f = 0$[3]. Our proof technique is inspired by the recent simpler proof of the $f + 1$ round consensus lower bound due to Moses and Rajsbaum [13], which uses forward induction and a new technique called *layering*.

Dwork and Skeen [5] give a related result for failure-free executions of non-blocking atomic commit. Their complexity analysis uses a different measure than the one we use here, but the two-round lower bound for failure-free executions implicitly appears in their paper. Lamport [11] presents a two-round lower bound for failure-free executions of uniform consensus in a model with only message omission failures.

2 The Lower Bound

A *state* of an algorithm (sometimes called global state or configuration), consists of value assignments to the local states of all the processes. If a process is failed, its local state consists of only one special symbol denoting that it is failed. Assume, for the lower bound of this section, that a *deterministic* uniform consensus algorithm is given. Given a state x of an execution of this algorithm, the state after one round is completely determined by two things: (1) which processes fail in this round, and (2) which of the messages sent by the failing processes are lost. An *environment* action consists of choosing these two things. Note: we assume that all the messages that are not lost in a given round arrive simultaneously, and therefore there is no non-determinism associated with the order in which messages arrive.

To obtain our impossibility result, it is sufficient to consider only a subset of all possible exe-

[2]The existence of two initial states that lead to different decision values is typically used to show that an initial bivalent state exists [7].

[3]Following the publication of our proof in [10], Charron-Bost and Schiper have extended the proof to cover the case $f = 0$.

cutions of the model, as generated by a subset of environment actions[4]. Thus, we define a *system* to be a subset of executions. Following [13], we consider a system in which at most one process fails in every round, and moreover, messages from the failed process are lost by a prefix of the processes, $\{p_1, \ldots, p_k\}$. We denote the set $\{p_1, \ldots, p_k\}$ by $[k]$, with $[0]$ denoting the empty set. We denote environment actions as follows: for $i > 0$, $(i, [k])$, denotes that p_i fails in this round and that messages sent from p_i to members of $[k]$ in this round will be lost, and only these messages will be lost. The action $(0, [0])$ models a round in which no failure occurs. Since we assume that at most t processes can crash, $t < n$, the action $(j, [k])$ for $j > 0$ is *applicable* to a state x only if fewer than t processes are failed at x, and process p_j is not failed at x. The action $(0, [0])$ is always applicable.

Given a state x and an action $(i, [k])$ that is applicable to x, we denote by $x \cdot (i, [k])$ the state after running the algorithm one round from x with the environment action $(i, [k])$. We denote: $\mathsf{L}(x) = \{x \cdot (i, [k]) | (i, [k]) \textit{ is applicable to } x\}$. We generalize this definition for a set of states X, as follows: $\mathsf{L}(X) = \cup_{x \in X} \mathsf{L}(x)$. We define L^k to be the application of L k times. Formally, L^k is defined recursively as follows: $\mathsf{L}^0(X) = X$; and $\mathsf{L}^k(X) = \mathsf{L}(\mathsf{L}^{k-1}(X))$.

Lower bound and impossibility proofs for distributed algorithms are typically based on a notion of *similarity* that captures the case that different states look the same to all but one processes. We define similarity as follows:

Definition 2.1 *States x and y are* similar, *denoted $x \sim y$, if they are identical, or if there exists a process p_j such that (a) x and y are identical except in the local state of p_j; and (b) there exists a process $p_i \neq p_j$ that is non-failed in both x and y (so, x and y are identical at least at p_i).*

A set X of states is similarity connected *if for every $x, y \in X$ there are states $x = x_0, \cdots, x_m = y$ such that $x_i \sim x_{i+1}$ for all $0 \leq i < m$.*

Given a state x, consider the execution extending x in which no failures occur after state x. Recall that in our model, an execution is fully determined by the initial state and the failure pattern. Since the algorithm solves uniform consensus, then all correct processes must decide upon the same value in this execution. We denote this decision value by $val(x)$.

We denote by $Init$ the set of initial states of the algorithm. No process is failed in an initial state. Thus, an initial state is determined by the inputs of the processes: for each combination of 0's and 1's there is an initial state in $Init$. We use the following well-known lemma (see, e.g., [13], for a proof in our context):

Lemma 2.1 *If $n > 1$, $Init$ is similarity connected.*

We now show that a uniform consensus algorithm cannot reach a decision in one round from some states in a similarity connected set.

Lemma 2.2 *Let X be a similarity connected set of states. Assume that there are states $x, x' \in X$ such that $val(x) \neq val(x')$. Assume that in every state in X there are at least three non-failed processes, and the number of failed processes is at most $t - 2$. Then, there is a state $y \in X$ such that in the state $y \cdot (0, [0])$ there is at least one correct process that has not decided.*

Proof: Since X is similarity connected, there are states $x = x_0 \sim x_1 \sim \cdots \sim x_m = x'$ in X. Since $val(x) \neq val(x')$, at some point along this chain there are two similar states $y, y' \in X$ such that $val(y) \neq val(y')$. Assume, by way of contradiction, that every correct process decides in both $y \cdot (0, [0])$ and $y' \cdot (0, [0])$. Without loss of generality, $val(y) = 1$, that is, all the processes decide 1

[4] Sets of environment actions are called *layers* in [13].

in $y \cdot (0,[0])$ and 0 in $y' \cdot (0,[0])$. The states y and y' are identical except in the local state of one process p_j, which implies that every process other than p_j that is non-failed in y is also non-failed in y', and vice versa. By assumption, there are at least two non-failed processes (other than p_j) in y and y'. Let p_m be the non-failed process, other than p_j, with the highest identifier in y and y', (that is, for all $k > m, k \neq j$: p_k is failed in y, y'). We distinguish between two cases:

Case 1: p_j is failed in one of these states, without loss of generality, y'. The states $y' \cdot (0,[0])$ and $y \cdot (j,[n])$ are identical because all the processes other than j have the same local states in y' and y, and no process hears from j in either extension. So, in $y \cdot (j,[n])$ all the non-failed processes decide 0. Consider the state $y \cdot (j,[m-1])$. There is at least one non-failed process p_i with $i < m$ which does not distinguish between $y \cdot (j,[n])$ and $y \cdot (j,[m-1])$, and therefore decides 0 in $y \cdot (j,[m-1])$. On the other hand, process p_m has exactly the same local state in $y \cdot (j,[m-1])$ as in $y \cdot (0,[0])$. Thus, p_m decides 1 in $y \cdot (j,[m-1])$. A contradiction to uniform agreement.

Case 2: p_j is non-failed in both y and y'. Since the number of failed processes in states in X is $\leq t-2$, we can extend y by failing two more processes, p_j and p_m. As argued above, p_m decides 1 in $y \cdot (j,[m-1])$. By uniform agreement, all the correct processes decide 1 in every extension of $y \cdot (j,[m-1])$, and in particular, in every extension of $y \cdot (j,[m-1]) \cdot (m,[n])$. By a symmetric argument for y', in any extension of $y' \cdot (j,[m-1]) \cdot (m,[n])$ all the correct processes decide 0. But no correct process other than p_m hears from p_j, so in $y \cdot (j,[m-1]) \cdot (m,[n])$ and $y' \cdot (j,[m-1]) \cdot (m,[n])$ all the correct processes have the same state, and by assumption, there is at least one correct process. A contradiction. ■

By plugging in *Init* for X in the above lemma, we get the lower bound for the case $f = 0$. In order to prove the general case, we now show that $\mathsf{L}^k(Init)$ is similarity connected for all $k \leq t$.

Lemma 2.3 *Let $X = \mathsf{L}^0(X)$ be a similarity connected set of states in which no process is failed, then $\mathsf{L}^k(X)$ is a similarity connected of states in which no more than k processes are failed for all $k \leq t$.*

Proof: By induction on k. The base case $k = 0$ is immediate. For the inductive step, assume that $\mathsf{L}^{k-1}(X)$ is similarity connected and in every state in $\mathsf{L}^{k-1}(X)$, at most $k-1 < t$ processes are failed.

We first show that for every $x \in \mathsf{L}^{k-1}(X)$, $\mathsf{L}(x)$ is similarity connected. For every process p_i that is non-failed at x, the states $x \cdot (i,[0])$ and $x \cdot (0,[0])$ are similar since they differ only in the local state of p_i. In addition, for every non-failed p_i and for all $l > 0$, the states $x \cdot (i,[l-1])$ and $x \cdot (i,[l])$ differ in the state of at most one process p_l, and are therefore similar.

It is left to show that if $x \sim x'$ (for $x \neq x'$), then there are similar states $y \in \mathsf{L}(x)$, $y' \in \mathsf{L}(x')$. To see this, recall that the states x and x' are identical except in the local state of one process p_i. Since the number of processes that are failed in x and x' is smaller than t, it is possible for p_i to fail unless it is already failed. If p_i is non-failed in both x and x', then the states $y = x \cdot (i,[n])$ and $y' = x' \cdot (i,[n])$ are identical, and therefore similar. If p_i is failed in one of these states, say x' (without loss of generality), then the states $x' \cdot (0,[0])$ and $x \cdot (i,[n])$ are identical, and hence similar. ■

Theorem 2.4 *Consider an algorithm for uniform consensus in the synchronous crash model with up to t failures, where $1 < t < n$. For every $0 \leq f \leq t-2$, there exists an execution of the algorithm with f failures in which it takes at least $f+2$ rounds for all the correct processes to decide.*

Proof: Fix f, $0 \leq f \leq t-2$. By Lemmas 2.1 and 2.3, $\mathsf{L}^f(Init)$ is similarity connected. By validity, there exist two states $x, x' \in Init$ so that $val(x) \neq val(x')$. Let y, y' be the states obtained by applying the action $(0,[0])$ f times to x, x', resp. Since y, y' are failure-free extensions of x, x', we get that $val(y) \neq val(y')$, and they are both in $\mathsf{L}^f(Init)$. Since $f \leq t-2$, and $t < n$, in every state of $\mathsf{L}^f(Init)$ there are at least three non-failed processes two of which can fail. Therefore, by Lemma 2.2, there is a state $z \in \mathsf{L}^f(Init)$, so that in the failure-free extension of z it takes at least two additional rounds for all the correct processes to decide. ■

3 The Bivalency Argument Does not Work

Bivalency is a widely-used technique for impossibility and lower bound proofs related to consensus, first introduced in [7]. A state is *bivalent* if two different executions starting at this state can lead to different decision values. Numerous proofs based on bivalency arguments exist in the literature, (see examples in [13, 1, 6]). Such proofs are based on the observation that a state in which some process has decided cannot be bivalent. Such a proof first shows that an initial bivalent state exists. It then shows, by induction, that there are executions that remain in a bivalent state (for a certain number of rounds in case of a lower bound proof), precluding decision. [6]

Recent simple proofs of the $f+1$ lower bound for consensus [1, 13] use bivalency arguments in the context of a system, i.e., a subset of executions. Let $\mathcal{S}$ be a system in which at most one process fails in each round. Those proofs, as well as ours, consider systems (ours and [13] strictly) contained in $\mathcal{S}$. A state of an algorithm is *bivalent with respect to* $\mathcal{S}$ if it can lead to different decision values in executions of $\mathcal{S}$.[5] The $\mathcal{S}BV$ *uniform consensus* problem replaces the validity condition with:

- *Bivalent validity with respect to system $\mathcal{S}$ ($\mathcal{S}BV$):* There is an initial state which is bivalent with respect to $\mathcal{S}$.

Remark: If we define bivalent validity as in [7] with respect to all executions, not just those in $\mathcal{S}$, then the $f+1$ lower bound does not hold, as shown in [4][6]. Since all bivalency proofs we know of use the validity property only in order to show that an initial bivalent state exists, such proofs also hold for the SBV version of their problems. We now prove Theorem 3.1, which shows that the $f+2$ lower bound of the previous section *does not* apply to $\mathcal{S}$BV uniform consensus.

Theorem 3.1 *There is an algorithm for $\mathcal{S}BV$ uniform consensus that has all the processes decide after one round in all failure-free executions.*

In Figure 1, we give an algorithm, Counter_Example, which solves $\mathcal{S}$BV uniform consensus and decides after one round in every failure-free execution. The algorithm uses a uniform consensus algorithm that satisfies (strong) validity (e.g., [2]), and tolerates up to $t = n-1$ failures. This algorithm is invoked by calling *Uniform_Consensus(v)*; we assume that if certain processes do not invoke the uniform consensus algorithm, other instances of the uniform consensus algorithm do not block, as they can assume these processes to be faulty after failing to receive messages from them.

It is easy to see that Counter_Example decides in one round in all failure-free executions. We now show that Counter_Example satisfies uniform agreement, termination, and $\mathcal{S}$BV validity. Termination is implied by termination of Uniform_Consensus, since all the correct processes either decide in Round 1 or run Uniform_Consensus.

To show uniform agreement, we first observe that if some process p_i decides 1 in Round 1, then all the processes that run uniform consensus have 1 as their input value for uniform consensus. This is because p_i does not send a Round 2 message, and thus, for any process that does execute

[5]Bivalency is explicitly defined with respect to system $\mathcal{S}$ in [13], and implicitly in [1].
[6]Many thanks to Bernadette Charron-Bost for pointing this out.

Function Counter_Example(v_i)
Round 1: Send a message to all the processes (including p_i).
  let S_1 be the set of processes from which Round 1 messages have been received.
  if $|S_1| = n$ **then return**(1) **fi**
Round 2: Send a message to all the processes.
  let S_2 be the set of processes from which Round 2 messages have been received.
  if $|S_2| < |S_1|$ **then** *init* $\leftarrow 1$ **else** *init* $\leftarrow 0$ **fi**
  return *Uniform_Consensus(init).*

Figure 1: A counter-example algorithm for bivalent validity.

Round 2, $|S_2| < |S_1|$ because $p_i \in S_1$ and $p_i \notin S_2$. By the (strong) validity property of uniform consensus, the only possible decision value in this case is 1. If no process decides in Round 1, then Uniform_Consensus ensures uniform agreement.

Finally, all the initial states of this algorithm are bivalent with respect to $\mathcal{S}$: If one process fails in Round 1 so that none of its messages are received, and there are no further failures, then the decision is 0. If there are no failures at all, then the decision is 1.

In summary, we observe that the bivalency argument, as used extensively in the literature (e.g., [13, 1, 6]), cannot show the lower bound proven in this paper, because in essence it relies on too weak a validity property. In this paper, we instead used a new technique, looking at a *pair* of states from which the *failure-free* executions lead to different decision values. Using this stronger notion, we constructed, for the first time, a proof by simple forward induction of the uniform consensus $f + 2$ round lower bound.

Acknowledgments
We thank Bernadette Charron-Bost, Alan Fekete, Rachid Guerraoui, Nancy Lynch, Keith Marzullo, and the referees for helpful comments that helped improve the presentation.

References

[1] M. K. Aguilera and S. Toueg. A simple bivalency-based proof that t-resilient consensus requires $t + 1$ rounds. *Inf. Process. Lett.*, 71(3-4):155–158, 1999.

[2] B. Charron-Bost and A. Schiper. Uniform consensus is harder than consensus (extended abstract). Technical Report DSC/2000/028, Swiss Federal Institute of Technology, Lausanne, Switzerland, May 2000.

[3] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, October 1990.

[4] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Inform. Comput.*, 88(2):156–186, October 1990.

[5] C. Dwork and D. Skeen. The inherent cost of nonblocking atomic commitment. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–11, 1983.

[6] F. Fich and E. Ruppert. Lower bounds in distributed computing. In *14th International Symposium on DIStributed Computing (DISC)*, pages 1–28, Oct. 2000.

[7] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32:374–382, April 1985.

[8] R. Guerraoui. Revisiting the relationship between non-blocking atomic commitment and consensus. In J.-M. Hélary and M. Raynal, editors, *9th International Workshop on Distributed Algorithms (WDAG)*, pages 87–100. Springer Verlag, September 1995. LNCS 972.

[9] R. Guerraoui. Indulgent algorithms. In *19th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 289–297, 2000.

[10] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults – a tutorial. Technical Report MIT-LCS-TR-821, MIT Laboratory for Computer Science, May 2001. Preliminary version in SIGACT News 32(2), pages 45–63, June 2001 (published May 15th 2001).

[11] L. Lamport. Lower bounds on consensus. Unpublished manuscript, March 2000.

[12] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[13] Y. Moses and S. Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002. Previous version in PODC 1998.

[14] A. Schiper. Early consensus in an asynchronous system with a weak failure detector. *Distributed Computing*, 10(3):149–157, 1997.

[15] D. Skeen. Nonblocking commit protocols. In *ACM SIGMOD International Symposium on Management of Data*, pages 133–142, 1981.