Simple Bivalency Proofs of the Lower Bounds in Synchronous Consensus Problems

Xianbing Wang, Yong-Meng Teo, and Jiannong Cao

Singapore-MIT Alliance E4-04-10, 4 Engineering Drive 3, Singapore 117576

***Abstract* — A fundamental problem of fault-tolerant distributed computing is for the reliable processes to reach a consensus. For a synchronous distributed system of n processes with up to t crash failures and f failures actually occur, we prove using a straightforward bivalency argument that the lower bound for reaching uniform consensus is $(f + 2)$-rounds in the case of $0 < f \le t - 2$, and a new lower bound for early-stopping consensus is min $(t + 1, f + 2)$-rounds where $0 \le f \le t$. Both proofs are simpler and more intuitive than the traditional methods such as backward induction. Our main contribution is that we solve the open problem of proving that bivalency can be applied to show the $(f + 2)$-rounds lower bound for synchronous uniform consensus.**

Index Terms — **consensus, synchronous distributed system, bivalency, and early-stopping.**

I. Introduction

Consensus is one of the fundamental problems in distributed computing theory and practice. Assuming that a distributed system consists of a set of n processes, $\{p_1, p_2, \ldots, p_n\}$, in the consensus problem, each process p_i initially proposes a value v_i, and all non-faulty processes have to decide on one common value v, in relation to the set of proposed values $V = \{ v_i \mid i = 1, \ldots, n \}$ [9, 12]. Without losing generality, we just consider $V = \{0, 1\}$. A process is *faulty* during an execution if its behavior deviates from that prescribed by its algorithm, otherwise it is *correct*. More precisely, the consensus problem is defined by the following three properties:

(1) T*ermination*: Every correct process eventually decides on a value.

X.B. Wang is with the Department of Computer Science, School of Computing, 3 Science Drive 2, National University of Singapore, Singapore 117543, and Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, and is on leave from Computer center, School of Computer, Wuhan University, China 430072 (e-mail: wangxb@comp.nus.edu.sg).

Y.M. Teo is with the Department of Computer Science, School of Computing, 3 Science Drive 2, National University of Singapore, Singapore 117543, and Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576 (e-mail: teoym@comp.nus.edu.sg).

J. Cao is with the Department of Computing, the Hong Kong Polytechnic University (csjcao@comp.polyu.edu.hk).

(2) *Validity*: If a process decides on v, then v was proposed by some processes.

(3) *Agreement*: No two correct processes decide differently.

The agreement property applies to only correct processes. Thus it is possible that a process decides on a distinct value just before crashing. The *uniform consensus* prevents such a possibility. It replaces the agreement property with the following:

(3') *Uniform Agreement*: No two processes (correct or not) decide differently.

Synchronous consensus protocols are based on the notion of *round* [1, 4, 10]. In a synchronous distributed system, every execution of the consensus protocol consists of a sequence of *rounds*. While in round r, each process executes sequentially the following steps:

(1) sends r-round messages to the other processes,

(2) waits for r-round messages from the other processes, and

(3) executes local computations.

Every process will start and finish the same *round* synchronously. Both message delay and relative processes speed are bounded and these bounds are known. The underlying communication system is assumed to be failure-free: there is no creation, alteration, loss or duplication of message.

Most existing synchronous consensus protocols are designed to tolerate crash failures [10]. 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. If a protocol allows processes to reach *consensus* in which at most t ($t < n - 1$) processes can crash, the protocol is said to tolerate t faults or to be a t-resilient consensus protocol. It has been proved in [1, 2, 4, 10] that the lower bound on the number of rounds is $t + 1$ for any synchronous consensus protocol tolerating up to t crash failures. If a protocol can achieve consensus and *stops* before round $t + 1$ when there are actually f ($f \le t$) faults, we call this an *early stopping* protocol. The well-known lower bound, min($t + 1$, $f + 2$) rounds, for early stopping consensus protocols in synchronous distributed systems has been proved [4]. Instead of considering the time at which process decide, we call those protocols which can achieve consensus before round $t + 1$ when there are actually f faults as *early-deciding* protocols. The

lower bound, (f + 1)-rounds, for early deciding consensus protocols in synchronous distributed system has been proved [3].

Bivalency is a standard technique for showing impossibility results and lower bounds related to consensus by forward induction. That is, there exists a state from which two different executions lead to different decisions. This was first introduced by [5] and used in [2] to show the lower bound for consensus. The proof is simpler and more intuitive than the traditional one; i.e., it uses simpler forward induction rather than a more complex backward induction, which in turn requires applying the induction hypothesis several times. But an open problem is how to extend this to prove the lower bound for early-deciding synchronous uniform consensus. As mentioned in [13] it is not immediately clear how to extend the proof in [2] to the case of uniform consensus. Keidar and Rajsbaum argue that the bivalency argument cannot be used to show the (f + 2)-rounds lower bound for the uniform consensus because in essence it relies on too weak a validity property [7].

Firstly, consider an algorithm for *uniform consensus* in the synchronous crash model with up to t failures, we show by applying the bivalency proof in [2] that for every $0 < f \leq t - 2$, 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. For $f = 0$ and $f = t - 1$, the lower bounds to reach uniform consensus are two rounds [6, 13] and (f + 1)-rounds [3] respectively. These have been proved and are not considered in this paper.

Secondly, we apply the bivalency proof to present a new lower bound proof for *early-stopping* synchronous consensus protocols. We show that in an execution with f failures, all correct processes decide and halt requires at least min (t + 1, f + 2) rounds, for every $0 \leq f \leq t$, where $t < n - 1$.

The rest paper is organized as follows. Section II describes the related work. Section III presents the bivalency argument. Based on the bivalency argument, we proved the uniform consensus lower bound in section IV, and present a new bivalency proof of early stopping consensus lower bound in section V. Our concluding remarks are in section VI.

II. Related Works

A. On Uniform Consensus

Charron-Bost and Schiper [3] proved that early-deciding uniform consensus protocols require at least $f + 2$ rounds whereas early-deciding consensus protocols require only f + 1 rounds if f is less than $t - 1$. For $f = t - 1$ or $f = t$, they showed that both consensus and uniform consensus require f + 1 rounds. Keidar and Rajsbaum [6, 7] used a different way to prove the same lower bound for synchronous early-deciding uniform consensus protocols and showed that for $1 < t < n$, every t-resilient uniform consensus protocol must perform two rounds in failure-free execution before all processes decide. In [13], a novel oracle argument is introduced to prove both lower bounds for synchronous consensus and uniform consensus. The underlying idea is: suppose there is a consensus algorithm A that can tolerate f faults and only executes f rounds of message exchange. Then another algorithm A' can be constructed that tolerates $f - 1$ faults and uses only $f - 1$ rounds. A' does so by making "oracle calls" to A. Repeating this process, an algorithm that only needs 0 rounds for 0 faults can be constructed, which is easily proven impossible.

But the proof in [3] proceeds by backward induction and therefore is difficult to follow. The oracle-based proof is fundamentally different from bivalency-based proofs [13]. The proof in [7] uses a new technique called layering [11] and proceeds by forward induction. Keidar and Rajsbaum argue that the bivalency argument cannot be applied to show the (f + 2)-rounds lower bound for uniform consensus. To support the argument, they changed the validity property to *Bivalent validity*[1] *with respect to system S* where at most one process fails in each round and defined *SBV* consensus to be the problem satisfying agreement, termination, and bivalent validity but not validity. They presented a counter example for *SBV* consensus in which all processes decide after one round in all failure-free executions but violate the validity property of consensus when all processes proposed 0.

However, in our opinion, the counter example only can support that bivalency argument can not prove the (f + 2)-rounds lower bound for synchronous uniform consensus in case of $f = 0$. It cannot show that there exists an algorithm which can solve the *SBV* consensus problem within f + 1 rounds, where $0 < f \leq t - 2$. In this paper, we present a bivalency proof for the (f + 2)-rounds lower bound for uniform consensus when $0 < f \leq t - 2$.

B. On Early Stopping Consensus

Lamport and Fisher discuss Byzantine general problem and transaction commit protocols in [8]. They prove a theorem that for a t-resilient protocol to solve the problem in a crash failure model, it needs at least t + 1 rounds. At the end of the paper, they point out that if only $f < t$ processes crash, no algorithm can avoid sending round f + 2 messages. They claim it can be proved and the proof is essentially the same as the previous proof of the theorem but is omitted. However, the proof of the theorem proceeds by complex backward induction and is difficult to follow.

Dolev, etc. discuss early-stopping in Byzantine agreement for synchronous distributed system [4]. First, they prove that there is no early-stopping protocol for simultaneous Byzantine agreement, which means the lower bound of simultaneous Byzantine agreement is (t + 1)-

[1] Bivalent Validity with respect to *S*: There is an initial state which is bivalent with respect to *S*.

rounds even actually only f crashes occur. Then they prove that min (t + 1, f + 2)-rounds is the lower bound for early-stopping eventual Byzantine agreement. A notion of c*ritical* edge is introduced to prove the theorem. An edge e in round k of history[2] H is critical if there is another history J, such that (a) J*'s* output is not equivalent to H[3]; (b) J is identical to H through round k except for edge e, and (c) J is the conservative extension of J_k. A history H is said to be *f-serial* if it has no more than f faults by round f + 1 and no process fails in H after round f + 1. The correct edge is used to construct contradiction with an *f-serial* history, but they do not use the bivalency arguments.

Charron-Bost and Schiper [3] prove the lower bound of early-deciding consensus is (f + 1)-rounds directly followed that t-resilient consensus protocol needs a least t + 1 rounds. They present an early deciding consensus protocol in which all correct processes can decide by the end of round f + 1, and point out some processes need send round f + 2 messages in some executions, where $f \le t - 2$. But they do not prove why these round f + 2 messages are needed.

III. The Bivalency Argument Proof

Bivalency argument proofs are based on the observation that a state in which some processes have decided cannot be bivalent. These proofs are based on a synchronous round-based system S with n processes and at most t crash failures such that at most one process crashes in each round. S is just a subset of executions of a consensus protocol.

A. Bivalency Argument Proof in [2]

Theorem 1 [2]. *Consider a synchronous round-based system 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 S.*

To prove the theorem, the following notations are introduced and used in this paper. A *configuration* of the system S is considered at the end of each round. Such a configuration is just the state of each process. Informally, a configuration C is 0-*valent* if starting from C the only possible decision value that correct processes can make is 0; it is 1-*valent* if starting from C the only possible decision value that correct processes can make is 1. C is *univalent* if it is either 0-valent or 1-valent; C is *bivalent* if it is not univalent. And 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 call a process *sink* in a partial run r_k if it sends no message and crashes at the beginning of round k.

The proof proceeds by contradiction as follows. Suppose there is an algorithm A that solves consensus in t rounds in S. Without loss of generality, assume that A is loquacious, i.e., at each round, every process is supposed to send a message to every process. First, Lemma 1 shows that in any run of A, the configuration at the beginning of round t must be univalent. Then a contradiction can be obtained by constructing a run of A that is bivalent at the beginning of round t in Lemma 3. This run is obtained by starting from a bivalent initial configuration in Lemma 2 and extending it one round at a time, while maintaining bivalency. Each one-round extension may require the killing of a process. Thus, the proof proceeds by proving three lemmas. The third Lemma contradicts the first and thus completes the proof of the theorem.

Lemma 1 [2]. *Any* (t − 1)*-round partial run* r_{t-1} *is univalent.*

Lemma 2 [2]. *There is a bivalent initial configuration.*

Lemma 3 [2]. *There is a bivalent* (t − 1)*-round partial run* r_{t-1}.

B. Extended Lemma

Lemma 4 will be used in the following proof.

Lemma 4. *For every bivalent k-round partial run* ($0 \le k \le t - 2$), r_k, *it can be extended by one round into a bivalent* (k + 1)*-round partial run*.

Proof. This lemma is proved as the induction step of Lemma 3. For readability, we present the detailed proof.

Assume, for contradiction, that every one-round extension of r_k is univalent.

Let r_{k+1}^* be the partial run extended from r_k by one round without crash in round k + 1. Then 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_{k+1}^0 of r_k that is 0-valent.

Note that r_{k+1}^* and r_{k+1}^0 must differ in round k + 1. 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_{k+1}^0 because in system S, at most one process crashes per round. Since p crashes in round k + 1 of r_{k+1}^0 it may fail to send a message to some processes, say to $\{q_1, q_2, \ldots, q_m\}$, where $0 \le m \le n$.

Starting from r_{k+1}^0, we now define (k + 1)-round partial runs $r_{k+1}^1, \ldots, r_{k+1}^m$ as follows. For every j, $1 \le j \le m$, r_{k+1}^j is identical to r_{k+1}^{j-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_{k+1}^j is univalent. There are two possible cases:

1. For all j, $0 \le j \le m$, r_{k+1}^j is 0-valent. So r_{k+1}^m and r_{k+1}^* are 0-valent and 1-valent respectively. The only difference between r_{k+1}^m and r_{k+1}^* is that p crashes at

[2] A *history* is an execution of the protocol.
[3] J and H make the different decision.

the end of round $k+1$ in r_{k+1}^{m}, 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 sinks at the beginning of round $k+2$. 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_{k+1}^{m} such that no process crashes after round $k+1$. But all correct processes decide 0 in r' because r_{k+1}^{m} is 0-valent — contradiction.

2. There is one j, $1 \le j \le m$, such that r_{k+1}^{j-1} is 0-valent while r_{k+1}^{j} is 1-valent. Extend r_{k+1}^{j-1} and r_{k+1}^{j} into partial runs, r and r', respectively, by crashing process q_j at the beginning of round $k+2$. 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' — contradiction. ❑

IV. Bivalency Proof of Uniform Consensus Lower Bound

Lemma 5. *Consider an early-deciding synchronous uniform consensus protocol, no process (correct or not) can decide in any bivalent partial run in S.*

Proof. Assume, for contradiction, a process p_i decides 1 in a bivalent partial run r_k. According to the definition of bivalent partial run, there is an execution continuing r_k, in which the final decision is 0. This is a violation of the uniform agreement property of uniform consensus. ❑

Theorem 2. *Consider a synchronous round-based system S with n processes and at most t crash failures that at most one process crashes in each round. If* $t < n$ *and* $0 < f \le t - 2$, *then there is no early-deciding protocol that solves uniform consensus in* $f+1$ *rounds in S.*

Proof. As in Theorem 1, the proof of Theorem 2 is also by contradiction. Assume the contrary, there is a protocol A that solves uniform consensus in $f+1$ rounds in S. That is, in any execution of A with f $(0 < f \le t-2)$ failures, all the correct processes must decide by the end of round $f+1$. Then, we prove that when f failures actually occur, all $(f+1)$-round partial runs extending from a bivalent $(f-1)$-round partial run, r_{f-1}, decide on the same value, it is contradiction to that r_{f-1} is bivalent. First, we introduce and prove Lemma 6, 7, and 8.

Lemma 6. *Any partial run* r_k $(k \le f+1)$ *of A without failure during round k in S is a univalent partial run.*

Proof. It is obviously to be true. If not, A cannot solve uniform consensus with f actual failures by the end of round $f+1$ because then we can construct at least $f+1$ consecutive bivalent partial runs by using Lemma 4 as in the following. When $k = f+1$, by Lemma 5, no process can decide by the end of round $f+1$.

Now consider $k < f+1$. According to the definition of S, at most $(k-1)$ processes crashed before round k. By Lemma 4, there are bivalent partial runs of A at each round from round $k+1$ to round $f+1$ because $f \le t-2$. Now there are $(f-k+1)$ rounds from round $k+1$ to $f+1$ and there are $f-(k-1)$ processes actually crash. Then by crashing one process in each round to construct a new bivalent partial run as extensions from r_k, the execution enters into a bivalent $(f+1)$-round partial round, in this case, by Lemma 5 no process can decide by the end of round $f+1$. – Contradiction. ❑

Lemma 7. *When extending from a bivalent f-round partial run,* r_f, *there exists only one type of bivalent partial runs of A in S, in which a process sank at the beginning of the round* $f+1$.

Proof. By Lemma 4, there exist bivalent $(f+1)$-round partial runs extended from r_f because $f \le t-2$. Assume the contrary, there exists a bivalent $(f+1)$-round partial run extended from r_f, r_{f+1}, in which process p_i received all messages in round $f+1$. By Lemma 5, p_i cannot make decision in round $f+1$ of r_{f+1}. But all correct process should decide by the end of round $f+1$ when actually f failures occur. According to the definition of S, there is no failure in round $f+1$ of the $(f+1)$-round partial run, r_{f+1}^{*}, which extended from r_f. Then, by Lemma 6, r_{f+1}^{*} is univalent. Now p_i cannot distinguish that it is in r_{f+1} or r_{f+1}^{*}, then p_i cannot make decision in round $f+1$ of r_{f+1}^{*} either. ❑

Lemma 8. *When extending from a bivalent f-round partial run,* r_f, *all univalent* $(f+1)$*-round partial runs make the same decision.*

Proof. From Lemma 7, if at least one process received all messages in a $(f+1)$-round partial run r_{f+1} which extended from r_f, then r_{f+1} is univalent. Without losing generality, assume all correct processes decide 1 in r_{f+1}^{*} where no failure occurs during round $f+1$. We use r_{f+1}^{k} to denote those partial runs that k processes do not received the message from the crashed process in round $f+1$ where $0 \le k \le n-f-1$.

Basis: Consider r_{f+1}^{0}, these partial runs are the same to r_{f+1}^{*} except one process crashes at the end of round $f+1$. Extend both r_{f+1}^{0} and r_{f+1}^{*} to round $f+2$ just by sinking one process in r_{f+1}^{*} which already crashed in r_{f+1}^{0}. Then two extensions are the same, they are univalent and will decide the same value definitely. According to the definition of univalent, all r_{f+1}^{0}s decide 1.

Hypothesis: *Suppose* $0 \le k < n-f-1$, *all* r_{f+1}^{k}*s decide* 1.

Induction Step: Now consider a univalent partial run, r_{f+1}^{k+1}, in which it differs from r_{f+1}^{k} by only one process, p_i, i.e., p_i received the message sent by the crashed process in round $f+1$ of r_{f+1}^{k}, but not in round $f+1$ of r_{f+1}^{k+1}. Extend both partial runs to round $f+2$ just by sinking p_i at the beginning of round $f+2$. Then, two extensions are the same and also are univalent to decide the same value definitely. Thus, all univalent r_{f+1}^{k+1}s decide 1.

By induction, all those univalent (f + 1)-round partial runs decide 1. ❑

Now, continue the proof of Theorem 2. By assumption, all correct processes decide by the end of round f + 1 when f failures actually occur.

By Lemma 2 and Lemma 4, there is a (f − 1)-round bivalent partial run r_{f-1}. Now extends it to round f. Consider r_f^* without failure occurs in round f, by Lemma 6, it is univalent. Without losing generality, assume r_f^* is 1-valent. Let r_f^k be those partial runs that k processes do not received the message from the crashed process in round f where $0 \le k \le n - f$, and r_{f+1}^{k*} denote extending (f + 1)-round partial run from r_f^k without failures in round f + 1. By Lemma 6, all those r_{f+1}^{k*}s are univalent.

Basis: Consider r_f^0, those partial runs are the same as r_f^* except that one process crashes at the end of round f. Extend both runs to round f + 1 just by sinking one process in r_f^* which already crashed in r_f^0. Then two extensions r_{f+1}^{0*} and r_{f+1}^* are the same, because r_f^* is 1-valent, r_{f+1}^* is univalent and will decide 1. Thus, all r_{f+1}^{0*}s decide 1.

Hypothesis: *Suppose* $0 \le k < n - f$, *all* r_{f+1}^{k*}*s decide* 1.

Induction Step: Consider two partial runs, r_f^k and r_f^{k+1}, where $0 \le k < n - f$, the partial runs differ by only one process, p_i, because p_i received the message sent by the crashed process in round f of r_f^k, but not in round f of r_f^{k+1}. By Lemma 6, r_{f+1}^{k*} and r_{f+1}^{k+1*} are univalent and r_{f+1}^{k*} is 1-valent by hypothesis.

Consider extending r_f^k and r_f^{k+1} to $r_{f+1}^{k'}$ and $r_{f+1}^{k+1'}$ respectively by crashing p_i that only p_j receives the message sent from p_i in both partial runs. Thus, $r_{f+1}^{k+1'}$ is the same as $r_{f+1}^{k'}$ except p_j. By Lemma 7, $r_{f+1}^{k'}$ and $r_{f+1}^{k+1'}$ are univalent because p_j received all messages in round f + 1. By Lemma 8, $r_{f+1}^{k'}$ is 1-valent because r_{f+1}^{k*} is 1-valent.

Now extending both $r_{f+1}^{k'}$ and $r_{f+1}^{k+1'}$ to round f + 2, $r_{f+2}^{k'}$ and $r_{f+2}^{k+1'}$, by sinking p_j at the beginning of round f + 2. Then, $r_{f+2}^{k'}$ and $r_{f+2}^{k+1'}$ are the same and also univalent. Thus, $r_{f+2}^{k+1'}$ decide 1 too and then $r_{f+1}^{k+1'}$ is 1-valent. By Lemma 8, r_{f+1}^{k+1*} must decide 1 too.

By induction, all (f + 1)-round partial runs without failure in round f + 1 decide 1. Because protocol A is also an f-resilient consensus protocol and we consider all possible extensions of r_{f-1}, but all those extensions decide the same at the end of round f + 1, then r_{f-1} must be univalent. – Contradiction. ❑

V. Bivalency Proof of Early Stopping Consensus Lower Bound

Lemma 9. *Consider an early-stopping synchronous consensus protocol, no correct process can decide and stop in any bivalent partial run in S.*

Proof. Assume, for contradiction, a correct process p_i decides 1 in round k of a bivalent partial run r_k and stop at the end of the round k. According to the definition of bivalent partial run, firstly, not all correct processes decide in this round, otherwise r_k is univalent; secondly, there is an execution continuing r_k, in which other correct processes decide 0. This is a violation of the agreement property of consensus. ❑

Theorem 3. *Consider a synchronous round-based system S with n processes and at most t crash failures that at most one process crashes in each round. If* $t < n - 1$ *and* $f \le t - 1$, *then there is no early-stopping protocol that solves consensus in* f + 1 *rounds in S.*

Proof. As in Theorem 1, the proof of Theorem 3 is also by contradiction. Assume the contrary, there is an early-stopping consensus protocol B that solves consensus in f + 1 rounds in S. That is, in any execution of B with f ($f \le t - 1$) failures, all the correct processes must decide and stop by the end of round f + 1. Then, we prove that when f failures actually occur, all (f + 1)-round partial runs extending from a bivalent f-round partial run, r_f, decide on the same value, it is contradiction to that r_f is bivalent. First, we introduce and prove Lemma 10.

Lemma 10. *Any partial run* r_k ($k \le f + 1$) *of B without failure during round k in S is a univalent partial run.*

Proof. It is obviously to be true. If not, B cannot solve consensus with f actual failures by the end of round f + 1 because then we can construct at least f + 1 consecutive bivalent partial runs by using Lemma 4 as in the following. When $k = f + 1$, by Lemma 9, no correct process can decide and stop by the end of round f + 1.

Now consider $k < f + 1$. According to the definition of S, at most (k − 1) processes crashed before round k. By Lemma 4, there are bivalent partial runs of B at each round from round k + 1 to round f + 1 because $f \le t - 1$. Now there are ($f - k$ + 1) rounds from round k + 1 to f + 1 and there are f − (k − 1) processes actually crash. Then by crashing one process in each round to construct a new bivalent partial run as extensions from r_k, the execution enters into a bivalent (f + 1)-round partial round, in this case, by Lemma 9 no process can decide and stop by the end of round f + 1. – Contradiction. ❑

Now, continue the poof of Theorem 3. By Lemma 2 and Lemma 4, for every $f \le t - 1$, there is a bivalent f-round partial run, r_f. Because there is at most one process, which can crash in each round in S and by Lemma 10, there must be a process crashed in each round of r_f, then there are $n - f$ processes in round f + 1 and one process may crash in the round. Let r_{f+1}^* be the (f + 1)-round partial run extended from r_f in which no failure occurs in round f + 1; Let r_{f+1}^k be those partial runs extended from r_f that k processes do not received the message from the crashed process in round f+1 where $0 \le k \le n - f - 1$.

By Lemma 9, r_{f+1}^* is univalent. According to the assumption, all correct processes must decide and stop in

r_{f+1}^{*}. Without losing generality, assume all correct processes decide 1 in r_{f+1}^{*}. Then all r_{f+1}^{m} must be univalent, where $0 \le m \le n - f - 2$. Otherwise, the processes which received all messages in both r_{f+1}^{*} and r_{f+1}^{m} cannot distinguish the two partial runs, by Lemma 9, they cannot decide and stop at the end of round $f + 1$ of r_{f+1}^{m}, thus they cannot decide and stop at the end of round $f + 1$ of r_{f+1}^{*} either (Contradiction to the assumption). Because in every r_{f+1}^{m}, those processes received all messages decide the same as they do in r_{f+1}^{*}, then every r_{f+1}^{m} will make the same decision as r_{f+1}^{*}. Thus, all correct processes decide 1 in every r_{f+1}^{m}.

Now, consider r_{f+1}^{n-f-1}s. Without losing generality, assume a r_{f+1}^{n-f-1}, r_{f+1}, is extended from r_f by sinking a process, q_0, at the beginning of round $f + 1$ and the alive processes are $q_1, \ldots, q_{n-f-1}$. There are $(n - f - 1)$ r_{f+1}^{1}s partial runs in which q_0 crashes in round $f + 1$. Let r_{f+1}^{1k} denote the partial run in which q_k does not receive the message from q_0, where $1 \le k \le n - f - 1$. It is obviously that q_k in both r_{f+1} and r_{f+1}^{1k} are same. Because in r_{f+1}^{1k} only q_k remains in the following rounds, the only decision it can make is 1. Then, all alive processes in r_{f+1} can decide 1 at the end of round $f + 1$, which means r_{f+1} is also 1-valent.

Thus, all extensions from r_f are 1-valent, it is contradiction to that r_f is bivalent. – Contradiction. ❑

Theorem 4. *The lower bound of early-stopping consensus protocols for synchronous distributed systems is min ($t + 1$, $f + 2$)-rounds, where $t < n - 1$ and $f \le t$.*

Proof. By Theorem 3, for $f < t$, the lower bound of early-stopping synchronous consensus protocols is $f + 2$ rounds. By theorem 1, for $f = t$, the lower bound is $t + 1$ rounds. Thus for $f \le t$, the lower bound of early-stopping synchronous consensus protocols is min ($t + 1$, $f + 2$)-rounds. ❑

VI. Conclusion

In this paper, we present new proofs of the lower bounds for both synchronous uniform consensus and early stopping synchronous consensus with crash failures by applying the bivalency argument. The proofs are by applying forward induction and are simple and intuitive than the traditional ones. Our main contribution is that we solve the open problem of applying the bivalency argument to show the $(f + 2)$-rounds lower bound for synchronous uniform consensus.

In our future work, we will investigate the properties of the bivalency argument and try to apply it in other areas of theoretical distributed computing.

References

1. H. Attiya, and J. Welch, "Distributed Computing: Fundamentals, Simulations and Advanced Topics", McGraw-Hill, 451 pages, 1998.
2. M.K. Aguilera, and S. Toueg, "A Simple Bivalency Proof that *t*-Resilient Consensus Requires $t + 1$ Rounds", Information Processing Letters, 71(3-4), 1999, 155-158.
3. B. Charron-Bost, and A. Schiper, "Uniform consensus harder than consensus", Technical Report DSC/2000/028, École Polytechnique Fédérale de Lausanne, Switzerland, May 2000.
4. D. Dolev, R. Reischuk, and R. Strong, "Early Stopping in Byzantine Agreement", J. ACM, vol. 37, no. 4, Apr. 1990, 720-741.
5. M. Fischer, N. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process", J. ACM, vol. 32, no. 2, Apr. 1985, 374–382.
6. I. Keidar and S. Rajsbaum, "On the Cost of Fault-Tolerant Consensus When There Are No Faults – A Tutorial", MIT Technical Report MIT-LCS-TR-821, May 24, 2001.
7. I. Keidar and S. Rajsbaum, "A Simple Proof of the Uniform Consensus Synchronous Lower Bound", Information Processing Letters, 85(1), 2003, 47-52.
8. L. Lamport, and M. Fischer, "Byzantine Generals and Transaction Commit Protocols", Technical Report, SRI Int'l, Apr. 1982.
9. L. Lamport, R. Sprocessak, and M. Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, Jul. 1982, 382-401.
10. N. Lynch, "Distributed Algorithms", Morgan Kaufmann, 1996.
11. Y. Moses and S. Rajsbaum, "A layered analysis of consensus", SIAM J. Comput., vol. 31, no. 4, 2002, 989-1021, Previous version in PODC 1998.
12. M. Pease, R. Sprocessak, and L. Lamport, "Reaching Agreement in the Presence of Faults", J. ACM, vol. 27, no. 2, Apr. 1980, 228-234.
13. J. Xu, "A Unified Proof of Minimum Time Complexity for Reaching Consensus and Uniform Consensus -- An Oracle-based Approach", *IEEE 21st Symposium on Reliable Distributed Systems (SRDS 2002)*, Osaka, Japan, October 2002.