A Unified Proof of Minimum Time Complexity for Reaching Consensus and Uniform Consensus – An Oracle-based Approach*

Jun Xu
College of Computing
Georgia Institute of Technology
jx@cc.gatech.edu

Abstract: In this paper, we offer new proofs to two lower bound results in distributed computing: a minimum of $f + 1$ and $f + 2$ rounds for reaching consensus and uniform consensus respectively when at most f fail-stop faults can happen. Here the computation model is synchronous message passing. Both proofs are based on a novel oracle argument. These two induction proofs are unified in the following sense: the induction steps are the same and only the initial step (f=0) needs to be proved separately. The techniques used in the proof offer new insights into the lower bound results in distributed computing.

Keywords: Consensus, uniform consensus, lower bounds, fault tolerance

1 Introduction

The problem of "reaching consensus under faults," as originally defined in [11] in the context of byzantine faults, is formulated as follows. A system consists of a set P of n isolated processors, among which at most f processors can be faulty. Each pair of processors may communicate with each other through a two-party message passing channel, using a medium that is reliable and of negligible delay (i.e., synchronous system model [9]). Here, two parties can implicitly infer the true identity of each other from the channel. Each processor p has a private value $\sigma(p)$. The problem is to design an algorithm in which each nonfaulty processor decides a value d such that (a) $d \in \{\sigma(p) : p \in P\}$ (validity), and (b) no two nonfaulty processors decide on different values (agreement). This problem is also referred to as f-resilient in the literature. "Reaching uniform consensus" is a stronger variation of the consensus problem (defined only in the context of fail-stop faults), which adds one more constraint: (c) if a faulty process does decide a value before they crash, it has to be the value that nonfaulty processors have agreed to.

There are two well-known lower bound results on reaching consensus and uniform consensus. One, due to Fischer and Lynch [4], states that at least $f + 1$ rounds of message passing are needed to reach consensus when at most f byzantine faults can happen. This bound is later extended to fail-stop faults by Dolev, Reischuk, and Strong [3]. The other, due to Charron-Bost and Schiper [2], states that at least $f + 2$ rounds are needed to reach uniform consensus when at most f fail-stop faults can happen. These two results are originally formulated and proved separately using different techniques.

In this paper, we offer a new, intuitive, and unified proof to both lower bounds, as compared to [4, 10, 1, 6, 7, 8] on reaching consensus, and [2, 7] on reaching uniform consensus. Our proof on the consensus is based on the following novel oracle argument. Suppose there is a consensus algorithm A that can tolerate f faults and only executes f rounds of message exchange. Then we can construct another algorithm A' that tolerates $f - 1$ faults and uses only $f - 1$ rounds. A' does so by making "oracle calls" to A. Repeating this process, we get an algorithm that only needs 0 rounds for 0 faults, which is easily proven impossible. The proof on the uniform consensus has exactly the same induction step (also using the oracle argument), and only the initial step needs to be proved separately.

Main contributions of this paper are summarized as

*The work was supported in part by the National Science Foundation under grant ITR/SY 0113933.

follows:

- Our proof is new and intuitive as compared to existing lower bound proofs on both reaching consensus [4, 10, 1, 6, 7, 8] and reaching uniform consensus [2, 7].
- To the best of our knowledge, the oracle argument is novel in the context of proving lower bounds for consensus and uniform consensus. The oracle argument simplifies the proof and offers new insights into the problem.
- Our proof framework can be adapted to proving other lower bound or impossibility results in distributed computing such as impossibility of reaching consensus under mobile failures [12], as explained at the end of the next section.

2 Our Proof

We follow the notations used in [11] and [4] as much as possible. Without loss of generality (shown in [4]), a distributed algorithm A for reaching consensus under fail-stop faults among a set of n processors $P = \{p_1, p_2, \cdots, p_n\}$, runs in a number of rounds (say t) as follows. During the first round, each nonfaulty processor p sends its private value $\sigma(p)$ to other processors. For $k \geq 1$, during the $(k+1)_{th}$ round, each nonfaulty processor sends every other processor a message containing information they have obtained in the k_{th} round, and each faulty processor may omit some of such messages during the round it crashes and will not participate in the protocol (become silent) in later rounds. After t rounds, each nonfaulty processor p_i runs a decision algorithm A_i that takes all the information the processor gathers during all rounds as input and output a value d_i. Consensus is reached if for any nonfaulty processors p_i, p_j, the following agreement and validity conditions hold: (a) $d_i = d_j = d$ and (b) d is the private value of some processors. Uniform consensus is reached if (c) a processor that decides before it crashes also decides on the consensus value reached by nonfaulty processors.

Since we are proving lower bound results, it makes perfect sense to assume the weakest capability for consensus and uniform consensus algorithms: binary private values (hence binary consensus value) and fail-stop faults. We extend the notation σ to denote a scenario function $\sigma : \bigcup_{i=1}^{\infty} P^i \rightarrow \{0,1\}$. For a processor p, $\sigma(p)$ still denotes $p's$ private value. In addition, for a string $w = q_i q_{i-1} \cdots q_1$ of processors, $\sigma(w)$ corresponds to the value which q_{i-1} told q_i that q_{i-2} told q_{i-1} ... that q_1 told q_2 was $q_1's$ private value. So if the consensus algorithm executes for k rounds, then σ is defined on $\{w : |w| \leq k+1\}$. We refer to this σ as a k-round scenario. Each scenario corresponds to an instance of program execution.

Throughout this paper, we will use p, x, y, z to denote faulty or nonfaulty processors, s to denote nonfaulty processors only, and w to denote a string of processors. Also, n will always denote the total number of processors. In a k-round scenario σ, a processor p is called nonfaulty if $\sigma(qpw) = \sigma(pw)$ for all $|w| \leq k-1$. On the other hand, if p is faulty at i_{th} ($i \leq k$) round, then the value $\sigma(qpw)$ ($|w| = i-1$) can be "omitting message in round i" and the value $\sigma(qpw')$ ($|w'| = j \geq i$) is "omitting message in round j." For each scenario σ and a processor p, we use σ_p to denote $p's$ view: $\sigma_p(pw) = \sigma(pw)$ and undefined elsewhere. Let Σ_p be the set of all t-round views at p. The consensus decision algorithm A_p at p, which runs t rounds, can be viewed as a *functional* (since σ_p is itself a function) $A_p : \Sigma_p \rightarrow \{0,1\}^n$.

We now introduce additional definitions and notations needed to prove our main theorems. In the following discussion, we always assume that a set of n processors $P = \{p_1, p_2, \cdots, p_n\}$ are involved in the algorithm. Each processor $p_i \in P$ runs the consensus decision algorithm A_i.

Definition 1. Let $S \subseteq P$ be a set of processors. We say that a scenario σ is *S-good* if every processor in S is nonfaulty in σ and all the faults are fail-stop faults. A view V at processor p is said to be *S-possible* if there exists an S-good scenario σ such that $V{=}\sigma_p$. A k-fault scenario is one in which *no more than* k processors are faulty, and all faults are fail-stop faults.

Definition 2. We say that a $(k+1)$-round scenario α is an extension of a k-round scenario β if $\alpha(w) = \beta(w)$ for $|w| \leq k+1$. Likewise, we say that a $(k+1)$-round view α_p is an extension of a k-round view β_p if $\alpha_p(pw) = \beta_p(pw)$ for $|w| \leq k$. Given two k-round ($k > 0$) scenarios α and β, we call the set $\{w : \alpha(w) \neq \beta(w)\}$ the *difference set* between α and β, denoted as $diff(\alpha, \beta)$.

Definition 3. Given a k-round S-possible view V at processor p, we call the following extension of V, denoted as V^{o+}, its *optimistic extension*: $V^{o+}(pqw)$:=$V(pw)$ for any $|w| = k$, if q has not been found to omit a single message in the view V, and $V^{o+}(pqw)$:="omitting message in round $k+1$" (q has fail-stopped) otherwise. It is clear from the definition that p can obtain V^{o+} from V through local computation.

Lemma 1. A k-round View V at processor p is S-possible $=>$ V^{o+} is S-possible.

Proof: Since V is S-possible, there is an S-good k-round scenario σ such that $V = \sigma_p$. Let $T \subseteq \overline{S}$ be the set of processors that omits no messages in the first $k-1$ rounds and omits no message to p during the k_{th} round. So if $x \in T$ is faulty, $x's$ fault must happen in round k. We modify scenario σ to make all processors in T nonfaulty by "forcing" them to send correct messages to all processors in round k. We refer to the resulting scenario as α. Note that this modification does not change $p's$ view (i.e., V=α_p). In scenario α, the set of nonfaulty processors is exactly $S \bigcup T$. Now we extend α to a $k+1$-round scenario β such that processors in $S \bigcup T$ remain nonfaulty in round $k+1$ and processors in $\overline{S \bigcup T}$ all remain *silent* in round $k+1$. Clearly β is S-good, and it is not hard to verify that $V^{o+} = \beta_p$. □

Definition 4. Given a k-round S-good scenario σ, we call the following extension of σ, denoted as σ^{g+}, its *S-good $\overline{S}$-fail extension*: $\sigma^{g+}(xpw) := \sigma(pw)$ for $p \in S$ and $|w| = k$, and $\sigma^{g+}(xpw)$:="omitting message in round $k+1$" for $p \in \overline{S}$ and $|w| = k$. Clearly σ^{g+} is S-good because every $s \in S$ continue to be nonfaulty in the $(k+1)_{th}$ round. Also, such an extension will not introduce any faults that are not fail-stop. For each processor p, we say that the view σ_p^{g+} is the S-good $\overline{S}$-fail extension of an S-possible view σ_p. Note that in general p can not obtain σ_p^{g+} from σ_p through local computation since it may not know who is faulty and who is not.

The following main theorem, combined with the well-known $f+1$-round algorithm [11] that assures consensus (even under byzantine faults), establishes the achievable lower bound of $f+1$ rounds when at most f faults can happen.

Theorem 2. (Consensus) There does not exists a consensus algorithm that can reach consensus in f rounds, when at most f faults can happen ($f \geq 0$). Here we assume that the total number of processors $n \geq f+2$.

Proof: We prove this by induction. The theorem holds for $f = 0$, since to reach consensus for 0 faults, at least one round is needed to convey the private value of each processor to others. Suppose the theorem holds for $f = k$, that is, there is no algorithm that can reach consensus under any k-fault scenario in k rounds. We prove the case for $f = k+1$ by contradiction. Suppose there is an algorithm $A =< A_1, A_2, \cdots, A_n >$ that can reach consensus in $k+1$ rounds under any $k+1$-fault scenarios. We construct an algorithm A' from A that reaches consensus in k rounds under any k-fault scenario. This contradicts our induction hypothesis.

Suppose no more than k faults can happen. Algorithm A' runs as follows. It first *simulates* A for the first k rounds. We refer to this induced k-round scenario as β. Let S be the set of nonfaulty processors in β ($|\overline{S}| \leq k$). Then without executing the $(k+1)_{th}$ round, A' lets each processor $x \in S$ decide on the value $A_x(\beta_x^{o+})$ and halt. Here β_x^{o+} is the optimistic extension of the view β_x (see Definition 3). We prove that the consensus is reached this way! To show this, note that A reaches consensus among S on the S-good $\overline{S}$-fail extension of β (β^{g+}), since β^{g+} is shown to be S-good in Definition 4. Now we claim that $A_i(\beta_i^{o+}) = A_i(\beta_i^{g+})$ for any $i \in S$, to be shown next in Lemma 3. So A reaches consensus on β^{o+} too. Since A' is a k-round algorithm, this finishes our proof. □

In proving the following lemma, we follow the notations used above. We first overview the ideas used in the proof. We would like to show that if the last round may contain up to one fault, having this round of knowledge will not help the processors reach consensus. In other words, if they can reach consensus with the aid of the fault-prone last round, they can reach consensus anyway without that round. To show this, we claim that the additional knowledge learned in the last round should not change the decision value based on the "optimistic extension". This is shown by contradiction. Suppose the new knowledge is going to cause a change in that decision value at a processor. The new knowledge basicly consists of all the messages sent by other processors, which are viewed as votes from these processors. So these votes are going to decide whether there will be a change in the deci-

sion value. Then we show that there must exist a tie-breaking vote, such that if this vote is missing due to a fault, no decision value can be made that guarantees consensus[1].

Lemma 3. (Florida Lemma) For every $i \in S$, $A_i(\beta_i^{o+}) = A_i(\beta_i^{g+})$.

Proof: We first show that $diff(\beta_x^{o+}, \beta_x^{g+}) \subseteq X \bigcup Y$, where $X = \{xsfw : s \in S, f \in \overline{S}, |w| = k-1\}$ and $Y = \{xfw : f \in \overline{S}, |w| = k\}$. We only need to show that $\beta_x^{o+}(w) = \beta_x^{g+}(w)$ for $w \in \overline{X \bigcup Y}$. To show this, we split $\overline{X \bigcup Y}$ into two subsets $J = \{w : |w| \leq k+1\}$ and $K = \{xspw : s, p \in S, |w| = k-1\}$. Then, for $w \in J$, $\beta_x^{o+}(w) = \beta_x(w) = \beta_x^{g+}(w)$. Also, for $xspw \in K$, $\beta_x^{o+}(xspw) = \beta(xpw) = \beta(pw) = \beta(spw) = \beta_x^{g+}(xspw)$. The first and the last equalities come from the definition of β_x^{o+} and β_x^{g+}, respectively.

Let $p_1, p_2, \cdots, p_l$ be an enumeration of the set $S - \{x\}$, where $1 \leq l = |S| - 1$. We split X into l disjoint subsets $X_1, X_2, \cdots$, and X_l, where $X_i = \{xp_ifw : f \in \overline{S}, |w| = k-1\}$, $i = 1, 2, \cdots, l$. In other words, X_i is what the nonfaulty processor p_i told x about all the faulty processors have told her in the k_{th} round. We induce $l+1$ auxiliary views of x, referred to as V_i, $i = 0, 1, 2, \cdots, l$, as follows:

$$\begin{cases} V_i(w) = \beta^{g+}(w) & : \quad w \in Y \bigcup (\bigcup_{j=1}^{i} X_j) \\ V_i(w) = \beta^{o+}(w) & : \quad otherwise \end{cases} \tag{1}$$

Note that $V_l = \beta_x^{g+}$ since $diff(\beta_x^{o+}, \beta_x^{g+}) \subseteq X \bigcup Y$. So V_l is S-possible. We claim that $V_i, i = l-1, l-2, \cdots, 0$ are all S-possible. We first prove that V_{l-1} is S-possible. Let α' be an S-good $k+1$-round scenario in which $V_l = \alpha'_x$. We modify α' to α in three places: (a) all processors in $\overline{S}$ become silent in round $k+1$ in α, (b) for each processor that starts to become faulty in round k, it is "forced" to say the same thing to p_l as it says to x in round k, and (c) what p_l tells other processors in round $k+1$ in α will be adjusted according to (b). Clearly processors in S remain nonfaulty in α and no faults are introduced in this modification that are not fail-stop. So α is S-good. It is not hard to verify that V_{l-1}=α_x. So V_{l-1} is S-possible. Similarly, we can prove that $V_i, i = l-2, l-3, \cdots, 0$ are all S-possible.

[1]Recount (one more round) in the Florida fashion may be needed to ensure consensus.

Now we claim that for $0 \leq i \leq l-1$, $A_x(V_i) = A_x(V_{i+1})$. Suppose not. There exists i such that $a = A_x(V_i) \neq A_x(V_{i+1}) = b$. Then we let σ and γ be S-good scenarios in which $V_i = \sigma_x$ and $V_{i+1} = \gamma_x$. Also we choose σ and γ such that every processor in $\overline{S}$ becomes silent in round $k+1$ (their existence implied in the previous paragraph). We modify scenario σ to scenario σ' in which p_{i+1} omits the message to x in round $k+1$ but sends correct messages to all other processors. We modify scenario γ to scenario γ' in exactly the same way. This makes p_{i+1} a faulty processor in both σ' and γ' (fail-stop fault). So both σ' and γ' are $k+1$-fault scenarios. Also, $x's$ view in both scenarios are the same, i.e., $\sigma'_x = \gamma'_x$, since σ and γ only differs on what p_{i+1} tells x in round $k+1$. We show that now x has a dilemma. Let $z \in S$, $z \neq x$, and $z \neq p_{i+1}$ ($z's$ existence guaranteed by $n \geq f+2$). We know that $A_z(\sigma_z) = A_x(\sigma_x) = a$. Also note that $\sigma_z = \sigma'_z$ since the modification from σ to σ' does not affect $z's$ view. Since the algorithm A reaches consensus for all $k+1$-round $k+1$-fault scenarios, including σ', we know that $A_x(\sigma'_x) = A_z(\sigma'_z)$. So $A_x(\sigma'_x) = A_z(\sigma'_z) = A_z(\sigma_z) = a$. Similarly we can show that $A_x(\gamma'_x) = A_z(\gamma'_z) = A_z(\gamma_z) = b$. So $a = A_x(\sigma'_x) = A_x(\gamma'_x) = b$, contradicting our hypothesis $a \neq b$.

Finally, it remains to prove that $A_x(\beta_x^{o+}) = A_x(V_0)$. Since β_x^{o+} is S-possible according to Lemma 1, we let α be an S-good scenario in which $\beta_x^{o+} = \alpha_x$. Then we modify α to α' such that all processors in $\overline{S}$ omit messages to x in round $k+1$. Clearly α' only contains fail-stop faults and is S-good. Also $V_0 = \alpha'_x$. Let $z \in S$ and $z \neq x$. Clearly $z's$ view does not change from α to α', i.e., $\alpha_z = \alpha'_z$. So $A_x(\beta_x^{o+}) = A_x(\alpha_x) = A_z(\alpha_z) = A_z(\alpha'_z) = A_x(\alpha'_x) = A_x(V_0)$.

Note that when $k = 0$, $X = \emptyset$ and the whole proof still works. □

Remark [Every Vote Counts]: The name of the Lemma comes from the fact that $\{\sigma_x^{g+}(w) : w \in X_i\}$, $i = 1, 2, \cdots, l$ can be viewed as "votes" from the $l = |S| - 1$ nonfaulty processors (other than x) $p_1, p_2, \cdots$, and p_l. Suppose $A_x(\sigma_x^{g+}) \neq A_x(\sigma_x^{o+})$. Then the inequality has to come from these "nonfaulty votes." The above proof shows that there exists a "tie-breaking vote" (from some X_i) which results in $A_x(V_i) \neq A_x(V_{i+1})$ (every nonfaulty vote counts). However, since the consensus depends on the

vote from X_{i+1}, it can be "destroyed" when p_{i+1} becomes faulty in the last round.

In the following, we assume that at most t processors can fail, $t < n$. Now f denotes the actual number of faults.

Theorem 4. (Uniform Consensus) There does not exists a consensus algorithm that can reach uniform consensus in $f + 1$ rounds, when up to f faults actually happen. Here $f \leq t - 2$.

Proof: The induction proof is almost identical to the proof of Theorem 2, except for the initial step. Like in Theorem 2, we again assume that a magic algorithm A exists that can reach uniform consensus under all f-fault scenarios in $f + 1$ rounds, when at most t processors can fail. Then if at most $f - 1$ faults actually happen and at most $t - 1$ processors can fail, we construct A', which simulates A for f rounds and induces a scenario β. Let S be the set of nonfaulty processors in β. For every processor p that has not yet crashed, A' lets it decide on the value $A_p(\beta_p^{o+})$ and halt. Lemma 3 again proves that the (usual) consensus can be reached this way. Since in the proof of Lemma 3, we need to introduce a potential additional fault, the assumption that at most $t - 1$ processors can fail is critical. Now we only need to show the uniform consensus part. Let $x \in S$. If a processor $y \in \overline{S}$ decides in round $a \leq f$ in β and becomes faulty in round a' ($a \leq a' \leq f$), then $A_y(\beta_y) = A_x(\beta_x^{g+})$, since β^{g+} is S-good and the magic algorithm A guarantees uniform consensus. Now $\beta_x^{g+} = \beta_x^{o+}$ according to Lemma 3 and β_x^{o+} is the consensus value among S. So A' guarantees uniform consensus too and uses only f rounds.

It remains to prove the initial step: at least two rounds are needed in certain scenarios where there are actually no faults (i.e., $f = 0$), when at most $t = 2$ faults may happen. Note that the assumption $f \leq t-2$ is clearly necessary here. We prove the initial step by contradiction. Suppose there is an algorithm A that reaches consensus in one round when no faults happen. Then if a processor x correctly receives messages from all the other processors in round 1, it must decide on a value. The reason is that $x's$ view could be a part of a fault-free scenario. If x does not decide a value in the first round and it turns out that the overall scenario is indeed fault-free, the consensus algorithm then runs for at least two rounds for this fault-free scenario.

Now let $p_1, p_2, \cdots, p_n$ be an enumeration of all the processors and $r_1, r_2, \cdots, r_n$ be their private values. Then in a scenario when no faults happen, the consensus value among all the processors is a deterministic function of $r_1, r_2, \cdots, r_n$, denoted as $f(r_1, r_2, \cdots, r_n)$. We note that $f(0, 0, \cdots, 0) = 0$ and $f(1, 1, \cdots, 1) = 1$ due to the validity requirement of the consensus.

Now consider the following scenario. Suppose a processor p_i becomes faulty in the first round, and sends a message only to another processor p_l. Since p_l has received messages from all other processors, it has to decide on a value as explained above. Now suppose p_l crashes during the second round and does not send out any message. Here we assume that all the other processors $\{p_j\}_{j \neq i,l}$ are nonfaulty in both rounds. Then, none of them ($\{p_j\}_{j \neq i,l}$) knows the value r_i but they all have to decide on the value $f(r_1, r_2, \cdots, r_i, \cdots, r_n)$, which p_l has decided on, due to the uniform consensus requirement. Clearly $f(r_1, r_2, \cdots, r_{i-1}, 0, r_{i+1}, \cdots, r_n) = f(r_1, r_2, \cdots, r_{i-1}, 1, r_{i+1}, \cdots, r_n)$ (*). Since in equality (*), i and $r_j, j = 1, 2, \cdots, n$ can be arbitrary, we get $0 = f(0, 0, \cdots, 0) = f(1, 0, \cdots, 0) = f(1, 1, 0, \cdots, 0) = \cdots = f(1, 1, \cdots, 1) = 1$, a contradiction. □

Remark: Using the same oracle argument and a variation of the Florida Lemma, we have also proved that reaching consensus is impossible in synchronous model when mobile failures can occur. The intuitive idea again is that if the algorithm is indeed "1-resilient" in the last round, it could have stopped one round earlier and obtained the consensus. However, we decide not to include the result in this paper since the proof is not tightly coupled with the proofs on consensus and uniform consensus lower bounds. However, it does demonstrate the wide applicability of the oracle argument and the Florida Lemma.

3 Related Work

The original lower bound proof [4] by Fischer and Lynch, assuming byzantine failures, is "monolithic" in nature and rather complex. Later version of the proof (assuming fail-stop faults) by Lynch [8] is also involved. Recently, simpler and more intuitive proofs have been proposed independently in [10] and [1] on fail-stop faults. In the fail-stop fault model, a non-

faulty processor never omits a message to other processors, while a faulty processor may omit some messages during the round in which fault happens and becomes *silent* (sending out no messages) in later rounds. Aguilera and Toueg's proof [1] is a forward induction based on Fischer's bivalency argument [5]. Our Florida Lemma argument (shown later) is similar to the logic used in that proof. However, our proof cleanly extracts the "oracle argument" out of the involved reasoning process, making the proof more intuitive and easier to understand. Also, our proof smoothly extends to the case of uniform consensus, while it is not immediately clear how to make such an extension in the proof of [1]. Moses and Rajsbaum [1] introduce a concept of layering combined with bivalency arguments to prove a set of impossibility results in both synchronous and asynchronous models. Using this layering concept, Keidar and Rajsbaum's tutorial [7] unifies the proof of three impossibility results in [7], two of which are the lower bounds for reaching consensus and reaching uniform consensus. However, though the layering argument is powerful in its ability to prove a large class of lower bound and impossibility results, it is not quite intuitive in nature. The most recent proof on the lower bound for reaching uniform consensus is due to Charron-Bost and Schiper [2]. The proof uses complicated induction steps, and is less intuitive compared to our oracle argument.

Our oracle-based proof on consensus is simple and intuitive because it proves by contradiction: construct one "absurd oracle" from another. It is fundamentally different from bivalency-based proofs [10, 1, 7], which require the construction of a bivalent state that is "preserved" from the first to the last round.

Lynch's *Distributed Algorithm* book [8] contains a proof of the consensus result that is not bivalency-based. In particular, *some* of the definitions introduced in our proof are related to definitions used in the proof in [8]. For example, the concept of $S-good\ \bar{S}-fail$ extension (V^{g+}) is related to the concept of *failure-free* run (ff) in [8]. However, the former is weaker and more restrictive than the latter.

The Florida Lemma also looks to certain extent similar to a Lemma (Lemma 6.35) used in the proof in [8] regarding *regular runs* (no more than k faults in first k rounds, $k = 1, 2, ...$). However, these two lemmas are actually different in three fundamental ways. First, Lemma 6.35 is existential: there must exist a chain of indistinguishable views between two *regular runs*. Florida Lemma, on the other hand, is constructive and stronger: V^{o+} and V^{g+} are evaluated to the same value by the local decision algorithm. Second, it is not clear whether Lemma 6.35 in [8] can be trivially extended to proving uniform consensus. The Florida Lemma, on the other hand, can be "reused" by the uniform consensus proof and after slight modification by the mobile (omission) failure proof. Finally, Florida Lemma and Lemma 6.35 are proving different results. Moreover, the Florida Lemma can not be trivially derived from any Lemmas used (Lemma 6.35 included) in the proof in [8].

4 Conclusion

In this paper, we presents a self-contained, simple, and intuitive proof to the time complexity for assuring consensus and uniform consensus under fail-stop faults, based on a novel oracle argument. Since oracle argument is not constructive in nature, it avoids the involved process of developing a counterexample (bivalency), which all existing proofs use. The oracle argument offers unique insights into the nature of the problem: when a more powerful algorithm (e.g., designed for $f+1$-fault scenarios) is used to deal with a weaker fault condition (e.g., f-fault scenarios), at least one round can be saved. This insight is proved as the Florida Lemma using the "every vote counts" argument. Since the oracle argument is shown to be a generic method, we expect to identify more applications of the oracle argument to proving lower bounds in distributed computing.

Acknowledgment

I would like to thank Drs. Richard J. Lipton and Yechezkel Zalcstein for encouraging me to publish this idea. I also thank Dr. Mustaque Ahamad and anonymous reviewers for their suggestions that help improve the quality and readability of the paper.

References

[1] M. Aguilera and S. Toueg. A simple bivalency-based proof that t-resilient consensus requires

t+1 rounds. *Information Processing Letters*, 71(3–4):155–158, 1999.

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

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

[4] M. Fischer and N. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, June 1982.

[5] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[6] V. Hadzilacos. A lower bound for byzantine agreement with fail-stop processors. Technical report, Technical Report 21–83, Department of Computer Science, Harvard University, Cambridge, MA, July 1983.

[7] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults - a tutorial. *SIGACT News*, 32(2):45–63, June 2001.

[8] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[9] N. Lynch and M. Fischer. On describing the behavior and implementation of distributed systems. *Theoretical Computer Science*, (13):17–43, 1981.

[10] Y. Moses and S. Rajsbaum. The unified structure of consensus: a layered analysis approach. In *17th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 123–132, June 1998. submitted for journal publication.

[11] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[12] N. Santoro and P. Widmayer. Time is not a healer. In *Proc. of ACM STOC'89*, pages 304–313, Paderborn, Germany, February 1999. Also in LNCS 349.