Lower Bounds for Achieving Synchronous Early Stopping Consensus with Orderly Crash Failures

Xianbing Wang[1], Yong-Meng Teo[1,2], and Jiannong Cao[3]

[1]Singapore-MIT Alliance, [2]Department of Computer Science, National University of Singapore,
[3]Department of Computing, Hong Kong Polytechnic University

***Abstract*—In this paper, we discuss the consensus problem for synchronous distributed systems with orderly crash failures. For a synchronous distributed system of n processes with up to t crash failures and f failures actually occur, first, we present a bivalency argument proof to solve the open problem of proving the lower bound, min (t + 1, f + 2) rounds, for early-stopping synchronous consensus with orderly crash failures, where t < n – 1. Then, we extend the system model with orderly crash failures to a new model in which a process is allowed to send multiple messages to the same destination process in a round and the failing processes still respect the order specified by the protocol in sending messages. For this new model, we present a uniform consensus protocol, in which all non-faulty processes always decide and stop immediately by the end of f + 1 rounds. We prove that the lower bound of early stopping protocols for both consensus and uniform consensus are f + 1 rounds under the new model, and our proposed protocol is optimal.**

***Index Terms*—Consensus, Orderly crash failure, Early stopping, Synchronous distributed system**

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 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 $V = \{ v_i \mid i = 1, \ldots, n \}$. Without losing generality, we just consider $V = \{0, 1\}$ in this paper. 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.

(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*. In a synchronous distributed system, every execution of the consensus protocol consists of a sequence of *rounds*. Every process will start and finish the same *round* synchronously. Both message delay and relative processes speed are bounded and these bounds are known. Most existing synchronous consensus protocols are designed to tolerate crash failures. 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 [8].

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 that the lower bound on the number of rounds is $t + 1$ for any synchronous consensus protocol tolerating up to t crash failures. The proofs can be found in [2][1][8].

If a protocol can achieve consensus and *stops* before round $t + 1$ when there are actually f ($f \le t$) crashes, we call it 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 [5]. If just consider the time at which processes decide, we call those protocols in which all processes decide before round $t + 1$ with actually f crashes as *early-deciding* protocols. The lower bound, ($f + 1$)-rounds, for early deciding synchronous consensus protocols has been proved [3]. For synchronous uniform consensus, the lower bound of $f + 2$ rounds for crash failures, where $f \le t - 2$, has been proved in [3][7][11], and for $f = t - 1$, [3] presents a synchronous uniform consensus protocol in which all processes decide by the end of round $f + 1$.

X. Wang is with Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576 (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, Hong Kong (csjcao@comp.polyu.edu.hk).

In this paper, we consider synchronous consensus protocols under the *orderly crash failure* model [5][9][10]. With orderly crash failures, the failing process must respect the order specified by the protocol in sending messages. That is, if a process fails to send a specified message, it must also fail to send any message specified to be sent after that message in the protocol ordering. In [5], the authors mentioned that they could not prove the lower bound of min $(t + 1, f + 2)$ rounds for the orderly crash failure model. So, our first contribution is to solve the open problem.

Then, we extend the system model with orderly crash failures to a new model in which a process is allowed to send multiple messages to the same destination process in a round and the failing processes still respect the order specified by the protocol in sending messages. For this new model, we present a uniform consensus protocol that tolerates up to t failures, in which all non-faulty processes always decide and stop immediately by the end of $f + 1$ rounds. Then we prove that the lower bound of early stopping protocols for both consensus and uniform consensus are $f + 1$ rounds under this new model, and show our proposed protocol is optimal.

In the paper, the lower bounds are proved using bivalency arguments. *Bivalency argument* is a technique that uses *forward induction* to show impossibility results and lower bounds that are related to consensus. It means that there exists a state from which two different executions lead to different decisions. The technique was first introduced by [6] and used in [1] to show the lower bound for achieving consensus simply and intuitively.

The rest of the paper is organized as follows. Section II describes the system and orderly failure model. Section III introduces the bivalency proof in [1] and revises it to work for the orderly crash failure model. Section IV presents the bivalency proof of lower bound for early-stopping synchronous consensus with orderly cash failures. Section V describes the new system and failure model. Section VI proposes a uniform consensus protocol for the new model, and presents its correctness proof. Section VII presents the bivalency proof for the lower bounds in the new model. Finally, section VIII concludes this paper.

II. Orderly Crash Failure Model

A distributed system consists of n processes, $\prod = \{p_1, \ldots, p_n\}$, that communicate and synchronize by sending and receiving messages. Each pair of processes, p_i and p_j, is connected by a channel. Both message delay and relative process speed are bounded, and these bounds are known. Every execution consists of a sequence of *rounds*. While in round r, each process executes the following steps sequentially:

(1) send round r messages to the other processes, but send at most one message to a destination process;
(2) wait for round r messages from the other processes;
(3) execute local computations.

Thus, the system is synchronous. The underlying communication system is assumed to be failure-free: there is no creation, alteration, loss or duplication of message.

As mentioned in the Introduction, we restrict the failure model to the *orderly crash failure* model. Figure 1 shows the model. Process p_j is specified to send messages m_1, .., m_i, .., m_n, in a round, to processes p_1, .., p_i, .., p_n, respectively. But it fails to send message m_i and stops by doing nothing. Then processes p_1, .., p_{i-1} must have successfully received m_1, .., m_{i-1} respectively, and processes p_i, .., p_n did not receive message from p_j in the same round.

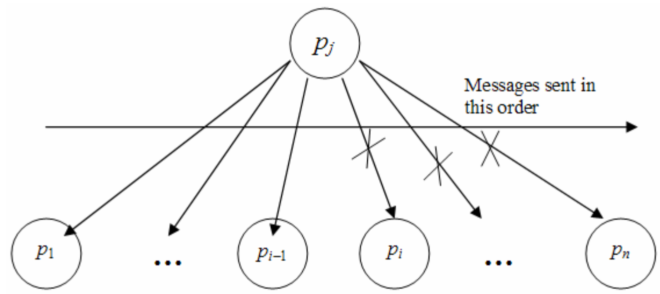

Figure 1. Example of orderly crash failures model

III. 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. The following notations are introduced and used in the bivalency argument proofs.

- *configuration*, a configuration of the system S is considered at the end of each round. Such a configuration is just the state of each process.
- 0-*valent*, a configuration C is 0-*valent* if starting from C the only possible decision value that correct processes can make is 0.
- 1-*valent*, a configuration C is 1-*valent* if starting from C the only possible decision value that correct processes can make is 1.
- *univalent*, C is univalent if it is either 0-valent or 1-valent.
- *bivalent*, C is bivalent if it is not univalent.
- *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.

- *same univalent*, two partial runs are *same univalent* if both are 1-valent or both are 0-valent.

We say that *a partial run r_k decides v* if all correct processes decide v by the end of round k of r_k. We say that a process *sinks* in a round if it sends no message and crashes at the beginning of the round.

The bivalency proof in [1] shows that a t-resilient consensus protocol requires $t + 1$ rounds in the crash failure model. We modify it to work for the orderly crash failure model in subsection *B*.

A. Bivalency Proof for Crash Failures

Theorem 1 [1]. *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.*

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, each process is supposed to send a message to every other process in a round. Three lemmas have been proved in [AT99]. The third Lemma contradicts the first and thus completes the proof of the theorem.

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

Lemma 2 [1]. *There is a bivalent initial configuration.*

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

B. Revised Proof for Orderly Crash Failures

To prove the same lower bound for the orderly crash failure model, some modifications to the proof of above Lemmas are needed. We show the revised proof works for the lower bound proof of the new model in section VII.

Theorem 1. *Consider a synchronous round-based system S with n processes and at most t* ***orderly 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.*

Proof. Suppose there is an algorithm A that solves consensus in t rounds in S.

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

Proof: Suppose there is a bivalent $(t - 1)$-round partial run r_{t-1}. Let r^* be the t-round partial run obtained by extending r_{t-1} by one round such that no process crashes in round t. Without loss of generality, we assume that all correct processes decide 0 in r^*. On the other hand, since the partial run r_{t-1} is bivalent, there exists one t-round partial run r^+ that extends r_{t-1} such that all correct processes decide 1. Note that in round t of r^+, exactly one process p must crash because in system S at most one process crashes per round.

Let $\{q_1, q_2, \ldots, q_m\}$ be the set of prescribed receivers of all orderly messages sent by p in round t. In partial run r^+, suppose q_{l-1} is the last process to which p delivers a message. Then, we can construct t-round partial runs r^j, $l \le j \le m$, based on run r^+ as follows. Let r^{l-1} be r^+. For every j, $l \le j \le m$, r^j is identical to r^{j-1} except that in r^j p sends a message to q_j before it crashes in round t. Every r^j is t-partial run and must be univalent. There are two possible cases:

Case 1. For each j, $l \le j \le m$, r^j is 1-valent. So r^m and r^* are 1-valent and 0-valent, respectively. The only difference between r^m and r^* is that p crashes at the end of round t in r^m, while p is correct in r^*. Every process except p in both partial runs maintains the same information, thus they cannot decide different in the two partial runs, —a contradiction.

Case 2. There is a j, $l \le j \le m$, such that r^{j-1} is 1-valent while r^j is 0-valent. Every process except q_j in both partial runs maintains the same information, and there is at least one other correct process because $n > t + 1$, then these processes cannot distinguish r^{j-1} and r^j, thus they cannot decide different in r^{j-1} and r^j, —a contradiction. ❑

Lemma 2. There exists a bivalent initial configuration.

Proof: Suppose all initial configurations are univalent. Consider two 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 sinking p at the beginning of round 1. ❑

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

In order to prove Lemma 3, we introduce and prove Lemma 4 first. Lemma 4 is used in the proofs in Section IV.

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

Proof. Assume, by contradiction, that every one-round extension of r_k is univalent. Because $k \le t - 2$, according to the definition of S, there are crashes which may occur till round $k + 2$.

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

Note that $r_{k+1}*$ and $r_{k+1}{}^+$ 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 $r_{k+1}{}^+$ because in system S at most one process crashes per round. Since p crashes in round $k + 1$ of $r_{k+1}{}^+$, it may fail to send a message to some processes. Suppose that, in round $k + 1$, the set of receivers of all orderly messages sent by p is $\{q_1, q_2, \ldots, q_m\}$, and q_{l-1} is the last process to which p delivers a message. Then p fails to send messages to $\{q_l, \ldots, q_m\}$.

Based on $r_{k+1}{}^+$, we now construct $(k + 1)$-round partial runs r_{k+1}^{l}, …, r_{k+1}^{m} as follows. Let r_{k+1}^{l-1} be r_{k+1}^{+}. For every j, $l \le j \le m$, 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 the round $k + 1$ of r_{k+1}^{j}. Recall the assumption that for every j, $l \le j \le m$, r_{k+1}^{j} is univalent. There are two possible cases:

Case 1. For all j, $l \le j \le m$, r_{k+1}^{j} are 1-valent. In this case r_{k+1}^{m} and $r_{k+1}*$ are 1-valent and 0-valent, respectively. The only difference between r_{k+1}^{m} and $r_{k+1}*$ is that p crashes at 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 $k + 2$ partial runs, r and r', extending from $r_{k+1}*$ and r_{k+1}^{m}, respectively. In r, process p sinks at the beginning of round $k + 2$ (before it sends any messages in that round). In r', there is no crash. Thus, partial runs r and r' are same. Since $r_{k+1}*$ is 0-valent and r_{k+1}^{m} is 1-valent, then r should be 0-valent but r' should be 1-valent —a contradiction.

Case 2. There is a j, $l \le j \le m$, such that r_{k+1}^{j-1} is 1-valent while r_{k+1}^{j} is 0-valent. Consider the $k + 2$ partial runs, r and r', extending from r_{k+1}^{j-1} and r_{k+1}^{j} by sinking process q_j at the beginning of round $k + 2$, and continuing with no additional crashes, respectively. Thus, partial runs r and r' are same. Since r_{k+1}^{j-1} is 1-valent and r_{k+1}^{j} is 0-valent, then r should be 1-valent but r' should be 0-valent —a contradiction. ❑

Lemma 5. *For every bivalent k-round partial run*, r_k, *if at least two processes may orderly crash in the following execution,* r_k *can be extended by one round into a bivalent* (k + 1)*-round partial run*.

Proof. It is obvious that there is possible crash in round k + 2. Thus the proof of Lemma 4 works for this Lemma. ❑

The proof of Lemma 3 then proceeds by forward induction:

Basis: By Lemma 2, there exists a bivalent initial configuration C_0. For $k = 0$, let r_0 be the 0-round partial run that ends in C_0.

Hypothesis: Suppose $0 \le k \le t - 2$. Let r_k be a bivalent k-round partial run.

Induction step: By Lemma 4, we can get a bivalent (k + 1)-round partial run r_{k+1}. ❑$_{\text{Lemma 3}}$

Lemma 3 contradicts Lemma 1. Thus, Theorem 1 must be true. ❑$_{\text{Theorem 1}}$

IV. LOWER BOUND FOR EARLY-STOPPING

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

Proof. The proof of Theorem 2 is by contradiction. Assume the contrary, there is an early-stopping protocol A that solves consensus in $f + 1$ rounds in S. That is, in any execution of A with f ($0 \le f \le t - 1$) failures, all the correct processes must decide and stop by the end of round $f + 1$. Follow the assumption, first, we introduce and prove Lemma 6, 7, 8, and 9.

Lemma 6. *For an early-stopping synchronous consensus protocol, no correct process can decide and stop in any bivalent partial run in S.*

Proof. Assume, by contradiction, a correct process p_i decides 1 and stop at the end of the round k of a bivalent partial run r_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. ❑

Lemma 7. *If a process decides v and stops in a round of a partial run, the partial run is v-valent.*

Proof. By Lemma 6, the partial run must be univalent. The process which decides and stops may be a correct process that it never crashes in following rounds. If the partial run is not v-valent, the agreement property is violated. ❑

Lemma 8. *Any partial run* r_k ($k \le f + 1$) *of A without failure during round k in S is univalent*.

Proof. If it is not univalent, A cannot solve consensus with f actual failures by the end of round $f + 1$, because we can construct at least $f + 1$ consecutive bivalent partial runs by using Lemma 5 as follows.

When $k = f + 1$, by Lemma 6, no 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. Now there are ($f - k + 1$) rounds from round $k + 1$ to $f + 1$ and there are $f - (k - 1)$ processes actually crash. And for every round j, where $k + 1 \le j \le f$, there are at least two processes may crash in the following rounds because of ($f \le t - 1$). Then, by Lemma 5, there are bivalent partial runs of A at each round from round $k + 1$ to round $f + 1$. We can crash 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 run. In this case, by Lemma 6 no process can decide and stop by the end of round $f + 1$. – Contradiction. ❑

Lemma 9. *When extending from a bivalent f-round partial run*, r_f, *all* ($f + 1$)*-round partial runs of A in S*, *in which at least one process receive all prescribed messages in round* $f + 1$, *are same univalent*.

Proof. First consider the ($f + 1$)-round partial run extended from r_f without failures, $r_{f+1}*$. By Lemma 8, $r_{f+1}*$ is univalent. According to the assumption, all processes decide and stop in round $f + 1$. Now consider another (f + 1)-round partial run extended from r_f with one orderly crash, r_{f+1}, in which process p_i received all prescribed messages in round $f + 1$. p_i cannot distinguish that it is in r_{f+1} or $r_{f+1}*$, if r_{f+1} is bivalent, then by Lemma 6, p_i cannot make decision and stop in round $f + 1$. This is a contradiction. Thus, r_{f+1} should be univalent also.

Without losing generality, assume $r_{f+1}*$ is 1-valent and all processes in $r_{f+1}*$ decide 1. Then p_i decides 1 and stops in round $f + 1$ of r_{f+1} also. Thus, r_{f+1} must be 1-valent, otherwise it violates the agreement property of consensus when p_i is a correct process. ❑

Now, continuing the proof of Theorem 2. By assumption, all correct processes decide and stop by the end of round $f + 1$.

Case 1: consider $0 < f \le t - 1$.

Let another protocol A' be the same as protocol A except A' is designed to tolerate up to f crash failures. By Lemma 2 and Lemma 4, there is an (f – 1)-round bivalent partial run r_{f-1} in protocol A'. It is clear that r_{f-1} is also an (f – 1)-round bivalent partial run in protocol A. We will prove that all executions extended from r_{f-1} in protocol A' decides the same according to the previous assumption.

Now first consider r_{f-1} in protocol A and extend it to round f. Consider r_f* without failure occurs in round f, by Lemma 8, 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 the $(f + 1)$-round partial run extending from r_f^k without failures in round $f + 1$. Because the messages sent from a process crash in order and total $n - f + 1$ processes remain in r_{f-1}, there are $(n - f + 1)$ r_f^ks for each k. By Lemma 8, all those $r_{f+1}{}^{k*}$s are univalent. We will prove they are same univalent as follows:

Basis: Consider r_f^0, those partial runs are the same as r_f* except that one process, p, crashes at the end of round f of r_f^0 but p have successfully delivered all its messages in the round. There are two cases: (1) some processes in both r_f^0 and r_f* deicide and stop in round f, by Lemma 7, r_f^0 should be 1-valent. (2) no such process exists, then extend both runs to round $f + 1$ just by sinking p in r_f* if p has not decided and stopped in round f of r_f*. 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: Because the messages sent from a process crash in order, then for every r_f^{k+1}, there exist one and only one r_f^k, where $0 \le k < n - f$, the partial runs differ by only one process, p_i, that 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 8, both $r_{f+1}{}^{k*}$ 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 9, $r_{f+1}{}^{k'}$ and $r_{f+1}{}^{k+1'}$ are univalent because p_j received all messages in round $f +$ 1, and $r_{f+1}{}^{k'}$ is 1-valent because $r_{f+1}{}^{k*}$ is 1-valent.

Because p_j will decide and stop at the end of round $f + 1$ as it does in $r_{f+1}{}^{k*}$ and $r_{f+1}{}^{k+1*}$, now extending both $r_{f+1}{}^{k'}$ and $r_{f+1}{}^{k+1'}$ to round $f + 2$, $r_{f+2}{}^{k'}$ and $r_{f+2}{}^{k+1'}$, without failure. Then, $r_{f+2}{}^{k'}$ and $r_{f+2}{}^{k+1'}$ are the same and also univalent. Thus, $r_{f+2}{}^{k+1'}$ is 1-valent as $r_{f+2}{}^{k'}$ and then $r_{f+1}{}^{k+1'}$ is also 1-valent. By Lemma 9, $r_{f+1}{}^{k+1*}$ must decide 1 too.

By induction, all $(f + 1)$-round partial runs, extended from r_{f-1}, without failure in round $f + 1$ decide 1. Because r_f* is univalent, then all $(f + 1)$-round partial runs extended from it are 1-valent.

Now consider protocol A'. Because it is the same as protocol A and is an f-resilient protocol, then when extended from r_{f-1} only one process can crash in round f or round $f + 1$. Thus, all $(f + 1)$-round partial runs extended from r_{f-1} are the same as discussed before in protocol A. But all those extensions eventually make the same decision by the end of round $f + 1$, then r_{f-1} is univalent. – Contradiction.

Case 2: consider $f = 0$.

Consider initial configurations C^0 and C^1 where all processes propose 0 in C^0 and all processes propose 1 in C^1. According to the validity property of consensus, C^1 is 1-valent and C^0 is 0-valent. Then all 1-round partial runs extended from C^0 are 0-valent and all 1-round partial runs extended from C^1 are 1-valent. Clearly, there are two initial configurations, C' and C'', that differ by the initial value of only one process p, such that the 1-round partial runs extended from C' and C'' without failure, r_1'* and r_1''*, decide different. Otherwise, both C^0 and C^1 will be the same v-valent, where v is 0 or 1, this violates the validity property of consensus. Without losing generality, assume that r_1'* is 1-valent and r_1''* is 0-valent.

By Lemma 9, when extended from any initial configuration C, all 1-round partial runs, in which at least one process receives all prescribed messages in round 1, are same univalent. Now consider a 1-round partial run, r_1', extended from C' and a 1-round partial run, r_1'', extended from C'', in both cases by crashing p and p only has successfully delivered its message to one process q in round 1. Then r_1' and r_1'' differ only by q. By Lemma 9, both r_1' and r_1'' are univalent and r_1' is 1-valent and r_1'' is 0-valent. Because all processes in r_1'* and r_1''* decide and stop in round 1, q will decide and stop in both r_1' and r_1''.

Now extending both r_1' and r_1'' to 2-round partial runs, r_2' and r_2'', without failures. Then r_2' and r_2'' are same, but r_2' is 1-valent and r_2'' is 0-valent —contradiction.

Thus, the Theorem 2 must be true. ❑

Theorem 3. *The lower bound of the early-stopping consensus protocols for synchronous distributed systems with orderly crash failures is min* $(t + 1, f + 2)$*-rounds, where* $t < n - 1$ *and* $f \le t$.

Proof. By Theorem 2, for $f < t$, the lower bound of early-stopping synchronous consensus protocols with orderly failures 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 with orderly crash failures is min $(t + 1, f + 2)$-rounds. ❑

V. New System Model

The new system also consists of n processes, $\prod=\{p_1, \ldots, p_n\}$, that communicate and synchronize by sending and receiving messages. Each pair of processes, p_i and p_j, is connected by a channel. The system executes protocols in a sequence of *rounds* and is still synchronous. While in round r, each process executes sequentially the following steps:

(1) send round r messages to the other processes. In this model, a process can send multiple messages to one destination process;

(2) wait for round r messages from the other processes;

(3) execute local computations.

Both message delay and relative process 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.

The failure model is similar to the orderly crash failures, the failing process must respect the messages sending order specified by the protocol.

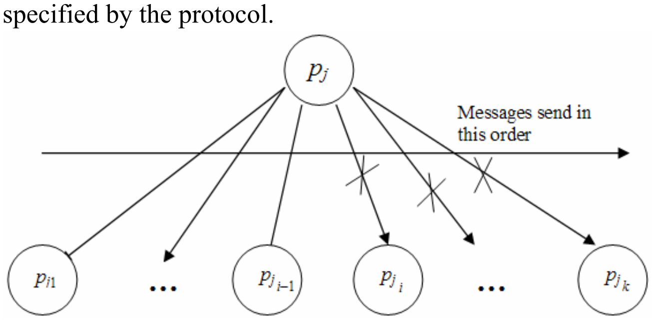

Figure 2. Example of the new model

Figure 2 shows the new failure model. Process p_j is specified to send messages, m_{j_1}, .., m_{j_i}, .., m_{j_k}, in a round, to processes, p_{j_1}, .., p_{j_i}, .., p_{j_k}, respectively, where k is the number of messages prescribed to be sent by p_j and $j_1, \ldots, j_i, \ldots, j_k \in N = \{1, .., n\}$. But it fails to send message m_{j_i} and stops by doing nothing. Then processes, p_{j_1}, .., $p_{j_{i-1}}$, must have successfully received m_{j_1}, .., $m_{j_{i-1}}$ respectively, and processes p_{j_i}, .., p_{j_k} did not receive message from p_j in the same round. It is obvious that p_j may send messages to the same process in a round.

The difference between the new model and the orderly crash failure model described in section II is that a process can set multiple messages to another process in a round in the new model. [10] presents a reference implementation of the new model by designing a similar protocol for synchronous reliable broadcast. How to realize the orderly crash in the new model is not focused in this paper. But we want to argue that the new model does not change the essence of the round notion, and in the crash failure model, allowing a process to send multiple messages to the same processes in a round does not improve the lower bound of early-stopping protocols.

VI. The Uniform Consensus Protocol

A. Protocol Description

Figure 3 presents our early stopping uniform consensus protocol for synchronous distributed systems in the new model, which can tolerate up to t ($t < n-1$) faults. Each process p_i is assigned a unique identity (ID) i ($1 \le i \le n$). Each process p_i invokes the function Consensus(v_i), where v_i is the value it proposes. It terminates with the invocation of the **return**() statement that provides the decided value.

```
Function Consensus(v_i)
r = 0;
while r < t + 1 do
   r = r + 1;
   if (i = r) then
      for j = r + 1 to n step 1 send (v_i) to p_j;
      for j = t + 1 to r + 1 step −1 send (v_i) to p_j;
   endif
   if (i = r) then return (v) at end of the round endif;
   let v be the value received from p_r during the rth round;
   if (i > r AND i ≤ t + 1) then      // processes in ∏CP
       if just receive one message sent by p_r  then v_i = v endif;
       if received two messages from p_r then return (v) endif;
   else      // processes in ∏non-CP
       if receive the message from p_r  then return (v) endif;
   endif
end while
```

Figure 3. The uniform consensus protocol

The protocol uses the *rotating coordinator paradigm*. Consensus() is made up of $t + 1$ rounds. Each round r ($1 \le r \le t + 1$) is managed by a predetermined *coordinator* p_r. Only the coordinator can send messages in a round, others just wait for message from the coordinator. Therefore, a round r consists of the following steps:

(1) The rotating coordinator in this round sends round r messages to the other processes.

(2) Every process waits for round r messages from the rotating coordinator in the round.

(3) After a process has received messages, it executes local computations.

Thus, all processes in the protocol are divided into two sets, $\prod_{CP}$ which consists of the IDs of the rotating coordinator processes, and $\prod_{\text{non-}CP} = \prod - \prod_{CP}$ which consists of the IDs of non-coordinator processes. Because the protocol aims to tolerate t process crashes and just consists of $t + 1$ rounds, the size of $\prod_{CP}$ is $t + 1$. For simplicity, we choose the first $t + 1$ processes from $\prod$ to form $\prod_{CP}$. Thus $\prod_{CP} = \{p_i \mid 1 \le i \le t + 1\}$.

During round r, the rotating coordinator p_r will first send v_r in ascending order to all processes whose IDs are larger than r, and then send v_r in descending order to processes whose IDs ranges from $t + 1$ to $r + 1$. The coordinator will decide on its own value at the end of the round. When a process p_j ($j \ne r$) in $\prod_{CP}$ receives a value from p_r in round r, it will set v_j to the received value; if it receives the value twice, it decides on the value and stops immediately. When a process p_j in $\prod_{\text{non-}CP}$ receives a value from p_r in the same round, it will decide on the received value and stop immediately.

B. Correctness Proof

Theorem 4. *The proposed protocol solves the Uniform Consensus problem in the new model, in which up to t processes can crash, and all non-faulty processes decide by the end of $f + 1$ rounds, where $t < n - 1$, $f \leq t$, and f is the number of failures that actually occur.*

Proof. It is obvious that the proposed protocol satisfies the Validity property.

To show that the Termination property is achieved, we first prove lemma 10.

Lemma 10. *If the rotating coordinator does not crash in a round, all non-faulty processes, which have not made decision, can make the same decision in the round.*

Proof. Assume the rotating coordinator p_r is the first coordinator that does not crash in its round. According to the protocol, all non-faulty processes in $\prod_{\text{non-}CP}$ that have not made decision, can receive a message from p_r in the round. Then they will decide on the value v_r maintained by p_r. All non-faulty processes in $\prod_{CP}$, except p_r, that have not made decision, can receive two messages from p_r in the round, then they will decide on v_r. p_r decide on its value, v_r, at the end of the round. ❑

Because at most t processes can crash, and actually f ($f \leq t$) processes crash, there is at least one of the first $f + 1$ rounds in which the corresponding coordinator does not crash. Assume r is the first round in which the coordinator p_r does not crash and r must be not more than $f + 1$. By Lemma 10, all non-faulty processes will decide in the round. Thus, the Termination property is achieved.

To prove that the Uniform Agreement property is achieved, we first prove Lemma 11 and Lemma 12.

Lemma 11. *If two processes decide in the same round, they make the same decision*.

Proof. According to the protocol, a process decides on the value of the current rotating coordinator in a round if it can do so. Thus all processes that decide in the same round must make the same decision. ❑

Lemma 12. *If all non-faulty processes in $\prod_{CP}$ maintain the same value v at the end of a round, all processes which decide after that round will make the same decision on v.*

Proof. This is obvious. During the following rounds a process decides on the value of the corresponding rotating coordinator. Because the values of the non-faulty processes in $\prod_{CP}$ are the same, all processes that decide after the round will make the same decision on v. ❑

Now, we prove the uniform agreement property, by contradiction, that two processes p_i and p_j ($i \neq j$) make different decisions. By Lemma 11, they must have not decided in the same round. Without losing generality, assume that p_i decides in round r and p_j decides in round r', and $r < r$'. There are two possible cases:

(a) p_i is the rotating coordinator p_r ($i = r$), by Lemma 10, all non-faulty processes will decide and terminate in round r. So p_j cannot decide in round r', — a contradiction.

(b) p_i is not the rotating coordinator p_r. According to the protocol and the property of the new model, when p_i decides on the value of p_r in round r, all non-faulty processes in $\prod_{CP}$ must have received at least one message from p_r and set their values to the value of p_r. So their values are the same. By Lemma 12, all processes which decide after round r will make the same decision on the value of p_r. Thus p_j should make the same decision as p_i, — a contradiction.

So, any two processes make the same decision. Thus, Theorem 4 must be true because all three properties of the uniform consensus are satisfied. ❑

VII. Lower Bounds for the New Model

In section III and IV we have proved that the lower bound for t-resilient protocols with the orderly crash failure model is $t + 1$ rounds in which any process can just send at most one message to one destination in a round, and the lower bound for early-stopping protocols is min ($t + 1, f + 2$) rounds, respectively. Now, consider the new model, in section VI we present a protocol which solves the early-stopping uniform consensus in $f + 1$ rounds. The lower bound of t-resilient consensus protocols will be less than $t + 1$ rounds if we can design a consensus protocol in the new model which can achieve consensus before round $f + 1$. However, we show that the proof of Theorem 1 also works for this new model. Thus, under the new model, the lower bound of t-resilient consensus protocols is still $t + 1$ rounds. Subsequently, we use this result to show that the lower bound of early stopping protocols for both consensus and uniform consensus is $f + 1$ rounds in the new model. Therefore, our proposed protocol is optimal.

A. Lower Bound for t-resilient Protocols

In this section, we adopt the notations and bivalency proof method in Theorem 1 and then analyze the proof of Theorem 1 and indicate that it also works for Theorem 5.

Theorem 5. *Consider a synchronous round-based system S* ***in the new model*** *with n processes and at most t 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.*

The proof of Theorem 5 proceeds by contradiction as follows. Suppose there is an algorithm A that solves consensus in t rounds in S. Like the proof of Theorem 1, three Lemmas are proved and the third contradicts the first one.

Lemma 13. *Any* $(t - 1)$*-round partial run r_{t-1} is univalent*.

The proof of Lemma 1 also works for this Lemma. Consider the set of receivers, $\{q_1, q_2, \ldots, q_m\}$, of all orderly messages sent by the crash process p in round t. In Lemma 1, q_i must be different than q_j, $i \neq j$, $1 \leq i, j \leq m$. But in the new model, q_i may be the same as q_j, $i \neq j$, $1 \leq i, j \leq m$. It is obvious that this does not affect the truth of the proof, because r^{j-1} and r^j still differ by only one process, q_j, in both models. Then except q_j, all other correct processes cannot distinguish r^{j-1} and r^j in both models. ❑

Lemma 14. *There is a bivalent initial configuration.*

The proof is the same as the proof of Lemma 2. ❑

Lemma 15. *There is a bivalent* $(t-1)$*-round partial run* r_{t-1}.

The proof of Lemma 3 also works for this Lemma. We just need show that the proof of Lemma 4 works under the new model. The reason is the same as the above in Lemma 13. Consider the set of receivers, $\{q_1, q_2, \ldots, q_m\}$, of all orderly messages sent by the crash process p in round $k + 1$. That the orderly set $\{q_1, q_2, \ldots, q_m\}$ and $\{q_l, \ldots, q_m\}$ have redundant processes does not affect the truth of the proof of Lemma 4, because $r_{k+1}{}^{j-1}$ and $r_{k+1}{}^{j}$ differ by only one process in both models. Then sink q_j at the beginning of round $k + 2$, the two $(k + 2)$-round partial runs extended from r^{j-1} and r^j are same in both models. ❑

Lemma 15 contradicts Lemma 13, thus Theorem 5 must be true. ❑Theorem 5

Corollary 1. *Consider a synchronous round-based system S* ***in the new model*** *with n processes and at most t failures such that at most one process crashes in each round. If n > t + 1 then there is no protocol that solves uniform consensus in t rounds in S.*

Proof. By the definitions of Agreement and Uniform agreement property and Theorem 5, it is obvious that the corollary is true. ❑

B. Lower Bound for Early Stopping Protocols

We now use the result of Theorem 5 to show that the lower bounds of early stopping protocols for both consensus and uniform consensus in the new model are $f + 1$ rounds.

Lemma 16. *Let A be a consensus protocol that tolerates up to t orderly crashes in the new model, where* $t < n - 1$. *Let f be the number of processes that actually fail. For each f,* $0 \leq f \leq t$, *there exists a run of A in which at least one process decides not earlier than round* $f + 1$.

Proof. Since $f \leq t$, the proof follows immediately from Theorem 5. ❑

Lemma 17. *Let A be a uniform consensus protocol that tolerates up to t orderly crashes in the new model. If* $t < n - 1$ *then for each f,* $0 \leq f \leq t$, *there exists a run of A in which at least one process decides not earlier than round* $f + 1$.

Proof. It follows immediately from Corollary 1. ❑

Theorem 6. *The lower bound for both early stopping consensus and early stopping uniform consensus protocols in the new model is* $f + 1$ *rounds*.

Proof. The proof is straightforward, following Lemma 16 and Lemma 17. ❑

We have demonstrated in Section VI that there exists an early stopping uniform consensus protocol for the new model, which achieves the lower bound of $f + 1$. By Theorem 6, our proposed protocol is optimal under the new model.

C. Discussion

One question is why the bivalency proof for the early-stopping lower bound in section IV cannot work for the new model. The reason is that when we assume an early-stopping protocol solve the consensus in the new model in $f + 1$ rounds (in fact we present one in section VI), Lemma 6, 7, 8, 9 are still true in this new model, but there are two problems which make the proof of Theorem 2 not workable for the new model:

First, for $f = 0$. In the proof of Theorem 2, there exists two initial configurations, C' and C'', that differ by the initial value of only one process p, but their 1-round failure free partial runs extensions, r_1'* and r_1''*, decide differently, r_1'* is 1-valent and r_1''* is 0-valent. For a 1-round partial run, r_1', extended from C' and a 1-round partial run, r_1'', extended from C'', in both runs, p crashed by only having successfully delivered its message to one process q in round 1. Then r_1' and r_1'' differ only by q. By Lemma 9, both r_1' and r_1'' are univalent and r_1' is 1-valent and r_1'' is 0-valent and q decides and stops in both r_1' and r_1''.

But in the new model, because p may send multiple messages to q in the round such as our proposed protocol, and no process can receive all prescribed messages sent to it in the round if p has just successfully sent one message, then the condition of Lemma 9, that one process received all prescribed messages, cannot be satisfied. Thus univalent of both r_1' and r_1'' cannot be ensured. Otherwise, if q gets all its messages from p, it cannot ensure only one process differ in both partial runs, because other processes may maintain different information in this case. Thus, the proof in Theorem 2 for $f = 0$ does not work for the new model.

Second: for $0 < f \leq t - 1$. In the proof of Theorem 2, when consider that r_f^k and r_f^{k+1} only differ by p_i and extend 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 9, $r_{f+1}{}^{k'}$ and $r_{f+1}{}^{k+1'}$ are univalent because p_j received all messages in round $f + 1$, and $r_{f+1}{}^{k'}$ is 1-valent because $r_{f+1}{}^{k*}$ is 1-valent.

But in the new model, by the same reason as above, the condition of Lemma 9 cannot be satisfied, because p_i may send multiple messages to p_j in the round like our proposed protocol and no process can receive all prescribed messages sent to it in the round. Thus univalent of both $r_{f+1}{}^{k'}$ and $r_{f+1}{}^{k+1'}$ cannot be ensured. The proof in Theorem 2 for $0 < f \leq t - 1$ does not work for the new model.

VIII. Conclusion

In this paper, we discuss the consensus problem for synchronous distributed systems with orderly crash failures. Our contributions are threefold. First, we present a bivalency argument proof to solve the open problem of proving the lower bound, min (t + 1, f + 2) rounds, for early-sopping synchronous consensus with orderly crash failures, where $t < n - 1$. Then, we extend the system model with orderly crash failures to a new model in which a process is allowed to send multiple messages to the same destination in a round and these messages are supposed to crash in order. For this new model, we present a uniform consensus protocol that tolerates up to t failures, in which all non-faulty processes always decide and stop immediately by the end of f + 1 rounds. Finally, we have proved that, under this new model, the lower bound of t-resilient consensus protocols is still t + 1 rounds; we then use this result to show that the lower bound of early stopping protocols for both consensus and uniform consensus are f + 1 rounds. As a result, our proposed protocol is optimal under this new model.

Acknowledgment

This work is partially supported by the National University of Singapore, under Academic Research Fund R-252-000-180-112.

References

[1] 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.

[2] H. Attiya, and J. Welch, “Distributed Computing: Fundamentals, Simulations and Advanced Topics”, McGraw-Hill, 451 pages, 1998.

[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] T. Chandra, and S. Toueg, “Time and Message Efficient Reliable Broadcasts”, 4th International Workshop on Distributed Algorithms, WDAG '90, Bari, Italy, September 24-26, 1990, Proceedings.

[5] D. Dolev, R. Reischuk, and R. Strong, “Early Stopping in Byzantine Agreement”, J. ACM, vol. 37, no. 4, Apr. 1990, 720-741.

[6] 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.

[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] N. Lynch, "Distributed Algorithms", Morgan Kaufmann, 1996.

[9] M. Merritt, "Notes on the Dolev-Strong lower bound for byzantine agreement", Unpublished manuscript, 1985.

[10] H. Tzeng, and K. Siu, "Message-Optimal Protocols for Fault-Tolerant Broadcasts/ Multicasts in Distributed Systems with Crash Failures", IEEE Transactions on Computers, vol. 44, no. 2, Feb. 1995, 346-352.

[11] 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.