INFORMATION AND COMPUTATION 91, 205–231 (1991)

Simultaneity Is Harder than Agreement*

BRIAN A. COAN

Bellcore, 445 South Street, Morristown, New Jersey 07960

AND

CYNTHIA DWORK

IBM Almaden Research Center,
650 Harry Road, San Jose, California 95120

We prove a strong lower bound on the number of rounds of message exchange required to achieve simultaneity (i.e., action in the same round) in certain synchronous fault-tolerant distributed systems. Specifically, our bound holds for any randomized protocol which solves either the simultaneous agreement problem or the distributed firing squad problem. It is known that any protocol that solves either of these problems and that is resilient to t processor faults has at least one execution that lasts at least $t+1$ rounds. We strengthen that bound by showing that all normal executions of such a protocol last at least $t+1$ rounds. The restriction to normal executions is a technical one that excludes certain executions in which a fortuitous pattern of processor faults enables early termination. The lower bounds proved in this paper contrast with known protocols that achieve agreement on a value (without simultaneity) in fewer than $t+1$ rounds in some normal executions. Our results are proved for randomized protocols, for a benign failure model (crash faults), and for a weak adversary. They apply *a fortiori* to deterministic protocols, more malicious failure models, and stronger adversaries.

1. INTRODUCTION

We prove a strong lower bound on the number of rounds of communication that any randomized protocol requires to solve either the simultaneous

* This work was performed while the two authors were at the Massachusetts Institute of Technology; it was supported by the Advanced Research Projects Agency of the Department of Defense under Contract N00014-83-K-0125, the National Science Foundation under Grant DCR-83-02391, the Office of Army Research under Contract DAAG29-84-K-0058, and the Office of Naval Research under Contract N00014-85-K-0168. A preliminary version of this paper appeared in the "Proceedings of the Fifth Symposium on Reliability in Distributed Software and Database Systems."

0890-5401/91 $3.00

agreement problem or the distributed firing squad problem. We prove our lower bound for a benign fault model, the crash fault model. Thus the bound also holds for more malicious fault models. In the crash fault model a faulty processor follows its protocol correctly for some time. Then, it stops completely (i.e., crashes) and sends no more messages. A correct processor follows its protocol thoughout an execution.

In this paper we only consider synchronous systems with reliable communication. A synchronous system is one in which communication takes place in a series of rounds. In each round each processor sends messages, receives all of the messages sent to it, and acts on the messages received. We only consider protocols which are designed to operate in a system of processors that communicate via a fully connected reliable message system. For some fixed t, we require that a protocol work correctly in any execution with at most t faulty processors. The protocol may produce arbitrary results in any excecution with more than t faulty processors.

For the problems that we consider it is known that any protocol that solves the problem has at least one execution that lasts at least $t+1$ rounds. In this paper we strengthen that bound by showing that all normal executions of such a protocol last at least $t+1$ rounds. An execution of a simultaneous agreement protocol is normal if the number of processors that crash by the end of round r is at most r for all $r \in \{1, ..., t\}$. For the distributed firing squad problem, there is a slightly different definition of a normal execution. We give this definition in Subsection 3.2.

Our lower bound applies to normal executions only. We believe that this limitation does not diminish the importance of our result. An adversary can ensure that all executions of a protocol are normal by limiting the rate at which processors crash. Our lower bound implies that a protocol can terminate before round $t+1$ only if a large number of processor crashes occur early in an execution. One would hope that such a pattern of failures would be rare in a reliable fault-tolerant distributed system. Thus, our lower bound removes much of the incentive to design protocols that terminate in fewer than $t+1$ rounds. An important special case of our lower bound is that every failure-free execution of a correct protocol must run for at least $t+1$ rounds.

Our approach is to prove lower bounds for extremely weak variants of the simultaneous agreement problem and the distributed firing squad problem. These are the lazy simultaneous weak agreement problem (defined in Subsection 2.1 of this paper) and the lazy distributed firing squad problem (defined in Subsection 3.1). These weak problem variants seem interesting only in the context of lower bounds. For example, each can be solved by a protocol that does nothing. The lower bound that we prove for weak variants holds a *fortiori* for stronger variants. It is for these stronger variants that one might wish to design a protocol.

In order to illustrate a natural application of our lower bounds, we give informal definitions of standard variants of the simultaneous agreement problem and the distributed firing squad problem. The lower bounds which we prove hold *a fortiori* for these problems.

In the *simultaneous agreement problem* each processor begins with an input chosen from a fixed set V. The objective is for all of the correct processors to agree on one element of V subject to the following four conditions.

Correctness Conditions for the Simultaneous Agreement Problem.

- *Agreement condition*: All correct processors that decide reach the same decision.
- *Validity condition*: If all correct processors start the protocol with input v then v is the decision of all of the correct processors that decide.
- *Simultaneity condition*: If any correct processor decides then all correct processors decide in the same round.
- *Termination condition*: The probability that all correct processors decide by round r tends to 1 as r tends to infinity.

In the *distributed firing squad problem* each processor may, at any round, receive a request to fire. This request comes from some unspecified source outside of the system of processors. We would like all of the correct processors to respond to this request by simultaneously firing (i.e., entering a distinguished state). In any protocol that solves this problem the processors must satisfy the following three conditions.

Correctness Conditions for the Distributed Firing Squad Problem.

- *Validity condition*: No correct processor fires unless some processor receives a request to fire.
- *Simultaneity condition*: If any correct processor fires then all correct processors fire in the same round.
- *Termination condition*: If any correct processor receives a request to fire then the probability that all correct processors fire by round r tends to 1 as r tends to infinity.

In this paper we formulate deterministic protocols as a special case of randomized protocols. The lower bound that we prove for randomized protocols holds *a fortiori* for deterministic protocols. In the preceding two problem statements we gave probabilistic termination conditions. Given our model, these conditions suffice for both randomized and deterministic protocols; however, if we know that a protocol is deterministic it is straightforward to replace the termination condition with an equivalent, standard termination condition that makes no mention of probabilities.

For example, the equivalent termination condition for the simultaneous agreement problem is that all correct processors eventually decide.

To facilitate the analysis of randomized consensus protocols, it is commonly assumed that there is some measure of independence between the failure modes of the system and the random choices of correct processors. We capture this assumption by imagining that the selection of which processors are faulty and the behavior of the faulty processors are under the control of an adversary that has specified, limited powers. A formal description of our adversary appears in Subsection 2.2. We prove our lower bound for a weak adversary. This contributes to the strength of the bound.

The consensus problems for which we obtain a strong lower bound require simultaneous (in the same round) action by all correct processors. Similar bounds seem to be impossible to obtain for consensus problems without a simultaneity requirement. Consider, for example, the eventual agreement problem, which is just the simultaneous agreement problem without the simultaneity requirement. There are many protocols for this problem that terminate in fewer than $t+1$ rounds in some normal executions; for example, the protocol of Chor and Coan (1985) has normal executions that terminate in 2 rounds for all t. We observe the following pattern: problems that require simultaneous action by the correct processors take at least $t+1$ rounds in all normal executions, but problems that merely require agreement on some value have protocols that terminate faster in some normal executions. This is why we say that simultaneity is harder than agreement.

The lower bounds of this paper hold only for problems where simultaneity is an absolute requirement. If we merely require simultaneity with high probability, then termination can be achieved in an expected number of rounds less than $t+1$. This can be done using known randomized agreement protocols and a technique first suggested by Rabin (1983).

We use an assortment of techniques to prove the various lower bounds given in this paper. For the lazy simultaneous weak agreement problem we use a standard technique (DeMillo, Lynch, and Merritt, 1982; Dolev, Reischuk, and Strong, 1990; Dolev and Strong, 1982; Dwork and Moses, 1990; Fischer and Lynch, 1982; Lamport and Fischer, 1982; Merritt, 1983; Merritt, 1984; Moses and Tuttle, 1988) to give a direct proof that there is no deterministic protocol that beats the bound. We then use a reduction to show that the bound holds for randomized protocols. Specifically, we show that if there is any randomized protocol that beats the bound, then there is a deterministic protocol that does the same. We prove our lower bound for the distributed firing squad problem by reducing the lazy simultaneous weak agreement problem to the lazy distributed firing squad problem. We use a different reduction from the one that Coan,

Dolev, Dwork, and Stockmeyer (1989) used to show a worst-case lower bound for the distributed firing squad problem. They used a reduction suitable for showing worst-case lower bounds. We use a new reduction suitable for showing a lower bound on all normal executions.

There has been a long history of work on lower bounds on the number of rounds required to solve various consensus problems in various fault models. The earliest lower bounds shown are for worst-case performance. Fischer and Lynch (1982) showed that, in the worst case, $t+1$ rounds are required to solve either the simultaneous or the eventual agreement problem in the Byzantine fault model. The lower bound was extended to the authenticated Byzantine fault model by DeMillo, Lynch, and Merritt (1982) and by Dolev and Strong (1982) and to the failure-by-omission fault model by Hadzilacos (1984). It was further extended to the crash fault model by Lamport and Fischer (1982). By reducing the agreement problem to the distributed firing squad problem, Coan, Dolev, Dwork, and Stockmeyer (1989) showed that in the worst case $t+1$ rounds are required to solve the distributed firing squad problem. These bounds are tight in the sense that there are matching protocols that terminate in $t+1$ rounds. Nevertheless, the bounds are weak in that they admit the possibility that there are many executions that terminate faster.

Dolev, Reischuk, and Strong (1990) were the first to investigate consensus protocols that sometimes terminate in fewer than $t+1$ rounds. They began by distinguishing between simultaneous agreement and eventual agreement. For both variants of the agreement problem they parameterized the lower bounds by f, the actual number of failures in a given execution of an agreement protocol. They showed that for all $f \in \{0, ..., t\}$ and for every protocol for simultaneous agreement there is at least one execution with f failures that lasts for at least $t+1$ rounds. They further showed that for any $f \in \{0, ..., t\}$ and for every protocol for eventual agreement there is at least one execution with f failures that lasts for at least $\min(f+2, t+1)$ rounds. Their results were for deterministic protocols only. The Dolev, Reischuk, and Strong lower bound is for the model with authenticated Byzantine faults. Their technique was extended to crash faults by Lamport and Fischer (1982).

In recent work Dwork and Moses (1990) and Moses and Tuttle (1988) have used the theory of knowledge (Halpern and Moses, 1990), to develop tight bounds on the number of rounds required to solve some consensus problems in all (including nonnormal) executions. Each paper restricts its attention to deterministic protocols. For deterministic protocols, each of the papers subsumes the lower bounds of this paper. The work of Dwork and Moses characterizes the crash fault model. The work of Moses and Tuttle characterizes various failure-by-omission fault models. Devising tight bounds for the Byzantine fault model remains an open question.

2. Lower Bounds for Agreement

In this section we develop a strong lower bound on the number of rounds that a randomized protocol requires to solve the lazy simultaneous weak agreement problem. We begin in Subsection 2.1 by defining this problem. In Subsection 2.2 we give the formal model in which this problem is solved. In Subsection 2.3 we review the known lower bound for deterministic protocols. Finally, in Subsection 2.4 we prove our lower bound for randomized protocols by showing that the existence of a randomized protocol that beats the bound implies the existence of a deterministic protocol that does the same.

2.1. *The Problem*

The *lazy simultaneous weak agreement problem* differs from the simultaneous agreement problem only in that the termination condition is deleted (making the problem lazy) and the validity condition is relaxed (making the problem weak). A protocol for this problem is run by a distributed system of n processors, at most t of which may be faulty. Each processor starts the protocol with an input value v from a fixed set V of legal inputs. Each processor may, at some point during the execution of the protocol, irrevocably decide on an element of V as its answer. There are three conditions that the processors must satisfy.

Correctness Conditions for the Lazy Simultaneous Weak Agreement Problem.

- *Agreement condition*: All correct processors that decide reach the same decision.
- *Validity condition*: If all processors are correct and start the protocol with input v, then v is the decision of all of the correct processors that decide.
- *Simultaneity condition*: If any correct processor decides then all correct processors decide in the same round.

2.2. *The Model*

We model an agreement protocol as a synchronous system of automata. Throughout this paper we let n be the number of processors in the system, we let $N = \{1, ..., n\}$, and we let $t \leqslant n-2$ be an upper bound on the number of processor faults that a protocol need tolerate. A protocol $\mathscr{P}$ is described by the following.

- D is the set of possible outcomes from the random choice performed by each processor each round.
- V is the input set.

• Q is the set of processor states for running (i.e., not crashed) processors. The state of a crashed processor is CRASHED $\notin Q$.

• $q_0 \in Q$ is the initial state in which each processor begins the protocol.

• M is the set of messages that can be sent by a running processor. The absence of a message (i.e., what is received from a crashed processor) is indicated by NULL $\notin M$.

• $\mu_{p,q}: \mathscr{I}^+ \times V \times D \times Q \to M$, for $(p, q) \in N^2$, is the message generation function for messages sent from running processor p to processor q. It is total function. ($\mathscr{I}^+$ denotes the set of positive integers.) The first component of the domain of $\mu_{p,q}$ is the current round number. The second is the input to processor p. The third is the current local random choice of processor p. The fourth is the current state of processor p.

• $\delta_p: (M \cup \{\text{NULL}\})^n \to Q$, for $p \in N$, is the state transition function for processor p. It is a total function. (The prior state of processor p is omitted from the domain of δ_p because it would be redundant. Processor p can send any required information in a message to itself.)

• $\gamma_p: Q \to \{\text{UNDECIDED}\} \cup V$, for $p \in N$, is the decision function for processor p. It is a total function. In the range of γ_p an element of V denotes a possible decision and UNDECIDED denotes the absence of a decision.

Each processor starts an execution of protocol $\mathscr{P}$ in the initial state q_0. The execution consists of a series of rounds. In each round each running processor makes a local random choice, sends messages to all processors, receives all messages sent to it this round, and makes a local state change. We assume *ordered sending*: the ith message sent by any running processor in any round is sent to processor i. In any execution of protocol $\mathscr{P}$ a correct processor behaves according to its transition rules during the entire execution. A faulty processor behaves according to its transition rules during some prefix of the execution, then it stops sending messages. The messages sent by a running processor depend on the current round, on its input, on its current local random choice, and on its current state.

For executions we use a succinct representation which contains enough information to determine the behavior of every processor in every round. Formally, an *execution* of protocol $\mathscr{P}$ is a triple (C, I, H), where C is a function from $N \times \mathscr{I}^+$ to D, where $I \in V^n$, and where H is a function from $N \times \mathscr{I}^+ \times N$ to {SENT, SILENT}. In an execution E we say that C is the *random-choice history*, I is the *input vector*, and H is the *message history*. For all $r \in \mathscr{I}^+$ and for all processors p, the value of $C(p, r)$ is the round r random choice of processor p. I is the vector of inputs to all of the processors. For all $r \in \mathscr{I}^+$ and for all processors p and q, the value of

$H(p, r, q)$ determines whether there is a round r message from processor p to processor q. If $H(p, r, q) = \text{SILENT}$ then processor p sends no round r message to processor q; otherwise, processor p sends the message that is specified by its protocol for its current state and current random choice. If $H(p, r, q) = \text{SENT}$ for all $(r, q) \in \mathcal{I}^+ \times N$ then processor p is *correct*; otherwise, processor p is *faulty*.

We impose two constraints on H. First we require that there be at most t faulty processors. Second, we model ordered sending with crash faults by requiring that, for all $(r, r') \in \mathcal{I}^+ \times \mathcal{I}^+$ and for all $(p, q, q') \in N^3$, the following hold: if $H(p, r, q) = \text{SENT}$ and $H(p, r', q') = \text{SILENT}$ then either $(1)\ r < r'$ or $(2)\ r = r'$ and $q < q'$. If $H(p, r, q) = \text{SENT}$ then we say that processor p is *running* for the qth message of round r; otherwise, we say that it has *crashed* by the qth message of round r. If a processor has crashed by the last message of round r, then we say that it has crashed by round r.

An execution $E = (C, I, H)$ is *normal* if for all $r \leqslant t$ at most r processors crash by round r.

We now give an inductive definition of the round r state of processor p in execution E of protocol $\mathcal{P}$, which we denote state(p, r, E). Assume that $E = (C, I, H)$ where $I = \langle i_1, ..., i_n \rangle$. We define state$(p, 0, E) = q_0$ and for all $r \in \mathcal{I}^+$ we define

$$\text{state}(p, r, E) = \begin{cases} \delta_p \langle m_1, ..., m_n \rangle & \text{if } H(p, r, n) = \text{SENT}; \\ \text{CRASHED} & \text{if } H(p, r, n) = \text{SILENT}, \end{cases}$$

where

$$m_q = \begin{cases} \mu_{q,p}(r, i_q, C(q, r), \text{state}(q, r-1, E)) & \text{if } H(q, r, p) = \text{SENT}; \\ \text{NULL} & \text{if } H(q, r, p) = \text{SILENT}. \end{cases}$$

Correct processor p *decides* v in round r of execution E if $\gamma_p(\text{state}(p, r, E)) = v$ and $\gamma_p(\text{state}(p, r', E)) = \text{UNDECIDED}$ for all $r' < r$. The running time of an execution is the number of rounds until the last correct processor decides.

When the performance of a randomized consensus protocol is analyzed, it is convenient to assume that the selection of which components fail and the behavior of the failed components are under the control of an adversary. Our lower bound is strong because we prove it for an extremely weak adversary. Specifically, our adversary selects the message history at the start of an execution without ever seeing either the inputs or the random choices. In contrast, a stronger adversary might be allowed to base its actions on the previous behavior (e.g., random choices) in an execution. Because it is weak, our adversary can be modeled as a message history. It

should be immediate that for any protocol $\mathscr{P}$, input vector I, random-choice history C, and adversary A there is a unique execution $E = (C, I, A)$.

Having defined our adversary, we can now define the expected running time of a fixed protocol $\mathscr{P}$. Let T be a random variable that, for a given execution of protocol $\mathscr{P}$, is the running time of the execution. For a fixed adversary A and input vector I, let the expected value of T, taken over the random choices, be denoted $E(T_{A,I})$. Define the expected running time for protocol $\mathscr{P}$ to be $\max_{A,I}(E(T_{A,I}))$.

We model deterministic protocols as a special case of randomized protocols. Specifically, a protocol is deterministic if $|D| = 1$, where D is the set of possible outcomes from the random choice performed by each processor each round. Whenever $|D| = 1$ we adopt the convention that $D = \{0\}$. We use $\mathbf{0}$ to denote the random-choice history that is 0 for all processors and for all rounds.

2.3. *The Lower Bound for Deterministic Protocols*

We state, without proof, a lower bound for deterministic protocols. The bound is a straightforward extension of a well-known result (DeMillo, Lynch, and Merritt, 1982; Dolev, Reischuk, and Strong, 1990; Dolev and Strong, 1982; Dwork and Moses, 1990; Fischer and Lynch, 1982; Lamport and Fischer, 1982; Merritt, 1983; Merritt, 1984; Moses and Tuttle, 1988). Its proof is given in Appendix A.

THEOREM 1. *Let $\mathscr{P}$ be any deterministic protocol that solves the lazy simultaneous weak agreement problem. In any normal execution of $\mathscr{P}$ no correct processor decides before round $t + 1$.*

2.4. *The Lower Bound for Randomized Protocols*

We prove that, for any protocol $\mathscr{P}$ that solves the lazy simultaneous weak agreement problem and for any normal execution E of protocol $\mathscr{P}$, no correct processor decides before round $t + 1$. We do this by showing how to transform an arbitrary protocol for the problem into a deterministic protocol for the same problem. Our transformation preserves the existence of normal executions that terminate in fewer than $t + 1$ rounds.

We define a function α that, given an arbitrary protocol for the lazy simultaneous weak agreement problem and an arbitrary random-choice history, produces a deterministic protocol for the lazy simultaneous weak agreement problem. Let protocol $\mathscr{P} = (D, V, Q, q_0, M, \mu, \delta, \gamma)$ be an arbitrary lazy simultaneous weak agreement protocol and let C be an arbitrary random-choice history. We define $\alpha(\mathscr{P}, C)$ to be the protocol $(\{0\}, V, Q, q_0, M, \mu', \delta, \gamma)$, where $\mu'_{p,q}(r, v, 0, s) = \mu_{p,q}(r, v, C(p, r), s)$ for all $(p, q) \in N^2$.

LEMMA 2. *Let* $\mathscr{P} = (D, V, Q, q_0, M, \mu, \delta, \gamma)$ *be any protocol that solves the lazy simultaneous weak agreement problem, let* C *be an arbitrary random-choice history, and let* $\mathscr{P}' = \alpha(\mathscr{P}, C)$. *Let* $E' = (\mathbf{0}, \langle i_1, ..., i_n \rangle, H)$ *be any execution of protocol* $\mathscr{P}'$. *If* $E = (C, \langle i_1, ..., i_n \rangle, H)$ *is an execution of protocol* $\mathscr{P}$, *then* $\text{state}(p, r, E') = \text{state}(p, r, E)$ *for all* $(p, r) \in N \times \{0, 1, ...\}$.

Proof. Suppose that $\mathscr{P}' = (\{0\}, V, Q, q_0, M, \mu', \delta, \gamma)$. The proof is by induction on r.

Basis ($r = 0$): It is immediate that $\text{state}(p, 0, E') = q_0 = \text{state}(p, 0, E)$.

Induction: If processor p crashes by round r in message history H then it is immediate that $\text{state}(p, r, E') = \text{CRASHED} = \text{state}(p, r, E)$. So, for the remainder of the proof we suppose that processor p does not crash by round r. Thus $H(p, r, n) = \text{SENT}$.

For all $(q, s) \in N^2$, let

$$m'_{q,s} = \begin{cases} \mu'_{q,s}(r, i_q, \mathbf{0}(q, r), \text{state}(q, r-1, E')) & \text{if } H(q, r, s) = \text{SENT}; \\ \text{NULL} & \text{if } H(q, r, s) = \text{SILENT}, \end{cases}$$

and let

$$m_{q,s} = \begin{cases} \mu_{q,s}(r, i_q, C(q, r), \text{state}(q, r-1, E)) & \text{if } H(q, r, s) = \text{SENT}; \\ \text{NULL} & \text{if } H(q, r, s) = \text{SILENT}. \end{cases}$$

We claim that $m'_{q,s} = m_{q,s}$ for all $(q, s) \in N^2$. If $H(q, r, s) = \text{SILENT}$ then it is immediate that the claim is true. If $H(q, r, s) = \text{SENT}$ then we calculate that

$$\begin{aligned} m'_{q,s} &= \mu'_{q,s}(r, i_q, \mathbf{0}(q, r), \text{state}(q, r-1, E')) \\ &= \mu_{q,s}(r, i_q, C(q, r), \text{state}(q, r-1, E)) \\ &= m_{q,s}. \end{aligned}$$

Having proved the claim that $m'_{q,s} = m_{q,s}$ for all $(q, s) \in N^2$, we now conclude the induction step by calculating that

$$\begin{aligned} \text{state}(p, r, E') &= \delta_p(m'_{1,p}, ..., m'_{n,p}) \\ &= \delta_p(m_{1,p}, ..., m_{n,p}) \\ &= \text{state}(p, r, E). \quad \blacksquare \end{aligned}$$

THEOREM 3. *Let* $\mathscr{P}$ *be any protocol that solves the lazy simultaneous weak agreement problem and let* C *be an arbitrary random-choice history. If* $\mathscr{P}' = \alpha(\mathscr{P}, C)$ *then protocol* $\mathscr{P}'$ *solves the lazy simultaneous weak agreement problem.*

Proof. The proof is by contradiction. Suppose that protocol $\mathscr{P}'$ does not solve the lazy simultaneous weak agreement problem. Then there is some execution $E = (\mathbf{0}, I, H)$ of protocol $\mathscr{P}'$ for which some correctness condition is violated. By Lemma 2, the same correctness condition is violated in execution (C, I, H) of protocol $\mathscr{P}$. This contradicts the assumption that protocol $\mathscr{P}$ solves the lazy simultaneous weak agreement problem. ▮

LEMMA 4. *Let $\mathscr{P}$ be any protocol that solves the lazy simultaneous weak agreement problem. Let $E = (C, I, H)$ be any normal execution of protocol $\mathscr{P}$ in which the correct processors decide in some round r. If $\mathscr{P}' = \alpha(\mathscr{P}, C)$ then $E' = (\mathbf{0}, I, H)$ is a normal execution of protocol $\mathscr{P}'$ in which the correct processors decide by round r.*

Proof. By assumption, execution E is normal. Executions E and E' have identical message histories. Therefore, execution E' is normal.

Having shown that execution E' is normal, we now show that all correct processors decide by round r in execution E'. By assumption, the correct processors decide by round r in execution E. By Lemma 2 and the fact that protocols $\mathscr{P}$ and $\mathscr{P}'$ have identical decision functions, correct processors decide in the same round in these two protocols. Therefore, the correct processors decide by round r in execution E' of protocol $\mathscr{P}'$. ▮

THEOREM 5. *Let $\mathscr{P}$ be any protocol that solves the lazy simultaneous weak agreement problem. Let E be any normal execution of protocol $\mathscr{P}$. The earliest round in which the correct processors can decide in execution E is round $t + 1$.*

Proof. The proof is by contradiction. Suppose that the correct processors decide before round $t + 1$ in execution E of protocol $\mathscr{P}$. Suppose $E = (C, I, H)$. Let $\mathscr{P}' = \alpha(\mathscr{P}, C)$. Protocol $\mathscr{P}'$ solves the lazy simultaneous weak agreement problem by Theorem 3. By Lemma 4, there is some normal execution of protocol $\mathscr{P}'$ in which the correct processors decide before round $t + 1$. By Theorem 1, such an execution is impossible, contradiction. ▮

It is an immediate corollary of Theorem 5 that $t + 1$ is a lower bound on the expected number of rounds required to solve the lazy simultaneous weak agreeement problem.

3. LOWER BOUNDS FOR DISTRIBUTED FIRING SQUAD

We develop a strong lower bound on the number of rounds required to solve the lazy distributed firing squad problem. In Subsection 3.1 we give

the correctness conditions for the lazy distributed firing squad problem. In Subsection 3.2 we give the formal model in which the lazy distributed firing squad problem is solved. In Subsection 3.3 we prove our lower bound on the number of rounds required to solve the lazy distributed firing squad problem. Our proof is by reducing the lazy distributed firing squad problem to the lazy simultaneous weak agreement problem.

It is common (Burns and Lynch, 1987; Coan, Dolev, Dwork, and Stockmeyer, 1989) to assume that processors that solve the distributed firing squad problem have no access to a global clock. We prove our lower bound for a system of processors that have access to a reliable global clock that indicates the current round number. This more powerful model strengthens our lower bound.

3.1. *The Problem*

A protocol for the *lazy distributed firing squad problem* is run by a distributed system of n processors, at most t of which may be faulty. Each processor may receive one or more request to fire during the execution of a protocol. Each correct processor may fire at any point during the execution of the protocol. There are two conditions that the correct processors must satisfy.

Correctness Conditions for the Lazy Distributed Firing Squad Problem.

- *Validity condition*: No correct processor fires unless some processor receives a request to fire.
- *Simultaneity condition*: If any correct processor fires then all correct processors fire in the same round.

3.2. *The Model*

We model a distributed firing squad protocol as a synchronous system of automata. We continue to follow the convention that n is the number of processors in the system, $N = \{1, ..., n\}$, and $t \leqslant n-2$ is an upper bound on the number of processor faults that a protocol need tolerate. A protocol $\mathscr{P}$ is described by the following.

- D is the set of possible outcomes from the random choice performed by each processor each round.
- Q is the set of processor states for running processors. The state of a crashed processor is CRASHED $\notin Q$.
- $q_0 \in Q$ is the initial state in which each processor begins the protocol.
- M is the set of messages that can be sent by a running processor. The absence of a message (i.e., what is received from a crashed processor) is indicated by NULL $\notin M$.

• $\mu_{p,q}: \mathscr{I}^+ \times \{\text{IDLE}, \text{REQUEST}\} \times D \times Q \to M$, for $(p, q) \in N^2$, is the message generation function for messages sent from running processor p to processor q. It is a total function. The first component of the domain of $\mu_{p,q}$ is the current round number. The second is equal to REQUEST if processor p receives a request in the current round and IDLE otherwise. The third is the current local random choice of processor p. The fourth is the current state of processor p.

• $\delta_p: (M \cup \{\text{NULL}\})^n \to Q$, for $p \in N$, is the state transition function for processor p. It is a total function.

• $\gamma_p: Q \to \{\text{UNDECIDED}, \text{FIRE}\}$, for $p \in N$, is the decision function for processor p. It is a total function. In the range of γ_p a decision to fire is denoted FIRE and the absence of such a decision is denoted UNDECIDED.

Each processor starts an execution of protocol $\mathscr{P}$ in the initial state q_0. The execution consists of a series of rounds. In each round each running processor makes a local random choice, possibly receives a request to fire, sends messages to all processors, receives all message sent to it this round, and makes a local state change. In any execution of protocol $\mathscr{P}$ a correct processor behaves according to its transition rules during the entire execution. A faulty processor behaves according to its transition rules during some prefix of the execution, then it stops sending messages. The messages sent by a running processor depend on the current round, on the presence or absence of a requiest to fire, on its current random choice, and on its current state.

For executions we use a succinct representation which contains enough information to determine the behavior of every processor in every round. Formally, an *execution* of protocol $\mathscr{P}$ is a triple (C, W, H), where C is a function from $N \times \mathscr{I}^+$ to D, where W is a function from $N \times \mathscr{I}^+$ to $\{\text{IDLE}, \text{REQUEST}\}$, and where H is a function from $N \times \mathscr{I}^+ \times N$ to $\{\text{SILENT}, \text{SENT}\}$. In an execution E we say that C is the *random-choice history*, W is the *request history*, and H is *the message history*. For all $r \in \mathscr{I}^+$ and for all processors p, the value of $C(p, r)$ is the round r random choice of processor p. For all $r \in \mathscr{I}^+$ and for all processors p, $W(p, r) = \text{REQUEST}$ if processor p receives a request to fire in round r and $W(p, r) = \text{IDLE}$ otherwise. Message histories for distributed firing squad protocols are identical to message histories for agreement protocols as described in Subsection 2.2. We use the same terminology and impose the same restrictions.

In execution $E = (C, W, H)$, round $l \geqslant 1$ is *quiescent* if $W(p, r) = \text{IDLE}$ for all $(p, r) \in N \times \{1, ..., l\}$. For any $l \geqslant 0$ execution $E = (C, W, H)$ is *l-normal* if for all $r \in \{l+1, ..., l+t\}$ at most $r - l$ processors crash by round r and either $l = 0$ or round l is quiescent and no processor crashes by round l. Execution E is *normal* if it is l-normal for some l. Intuitively an l-normal execution consists of l rounds in which nothing interesting happens

followed by a suffix execution analogous to the normal executions defined for agreement.

We now give an inductive definition of the round r state of processor p in execution $E=(C, W, H)$ of protocol $\mathscr{P}$, which we denote state(p, r, E). We define state$(p, 0, E)=q_0$ and for all $r \in \mathscr{I}^+$ we define

$$\text{state}(p, r, E)=\begin{cases}\delta_p\langle m_1, \ldots, m_n\rangle & \text{if } H(p, r, n)=\text{SENT;}\\ \text{CRASHED} & \text{if } H(p, r, n)=\text{SILENT,}\end{cases}$$

where

$$m_q=\begin{cases}\mu_{q,p}(r, W(q, r), C(q, r), \text{state}(q, r-1, E)) & \text{if } H(q, r, p)=\text{SENT;}\\ \text{NULL} & \text{if } H(q, r, p)=\text{SILENT.}\end{cases}$$

A correct processor p *fires* in round r of execution E if $\gamma_p(\text{state}(p, r, E))=\text{FIRE}$ and $\gamma_p(\text{state}(p, r', E))=\text{UNDECIDED}$ for all $r'<r$. We measure the running time of an execution of a randomized distributed firing squad protocol as the number of rounds that elapse until the last correct processor fires.

Our adversary and our method of calculating the expected cost of a distributed firing squad protocol are the obvious analogues of the adversary and method given for agreement protocols at the end of Subsection 2.2. We model deterministic distributed firing squad protocols in the same way that we modeled deterministic agreement protocols.

3.3. *The Lower Bound*

In this subsection we prove that all l-normal executions of a lazy distributed firing squad protocol take at least $l+t+1$ rounds. We do this by reducing the lazy simultaneous weak agreement problem to the lazy distributed firing squad problem. We use a different reduction from the one that Coan, Dolev, Dwork, and Stockmeyer (1989) used to show worst-case bounds for the distributed firing squad problem. Using our new reduction, we prove that if there is any l-normal execution of a lazy distributed firing squad protocol in which the correct processors fire before round $l+t+1$ then there is a normal execution of a lazy simultaneous weak agreement protocol in which all correct processors decide before round $t+1$. Because there are no normal executions of a lazy simultaneous weak agreement protocol in which the correct processors decide before round $t+1$ (Theorem 5), we conclude that there are no l-normal executions of a lazy distributed firing squad protocol in which the correct processors decide before round $l+t+1$.

We begin with an informal description of our reduction. We use an arbitrary lazy distributed firing squad protocol $\mathscr{P}$ as a basis for con-

structing a lazy simultaneous weak agreement protocol $\mathscr{P}'$. Protocol $\mathscr{P}'$ has input set $\{1, 2\}$. In protocol $\mathscr{P}'$ two copies of protocol $\mathscr{P}$ are run in parallel. One copy corresponds to input 1, and the other corresponds to input 2. In protocol $\mathscr{P}'$ an arbitrary processor p decides in the earliest round in which it would fire in either of the simulated copies of $\mathscr{P}$. If one copy of $\mathscr{P}$ fires first then processor p decides on the value that corresponds to that copy. If both copies fire together then processor p decides 1. There are two components to each random choice made by processor p in protocol $\mathscr{P}'$. One component is used to provide random choices to the two simulated copies of $\mathscr{P}$. The other component is used to provide requests to fire to the simulated copy of $\mathscr{P}$ that corresponds to the input to processor p. Processor p gives no requests to the other simulated copy of $\mathscr{P}$. A formal description of this reduction follows.

We define a function β that, given an arbitrary protocol for the lazy distributed firing squad problem, produces a protocol for the lazy simultaneous weak agreement problem. Let protocol $\mathscr{P} = (D, Q, q_0, M, \mu, \delta, \gamma)$ be a arbitrary lazy distributed firing squad protocol. We define $\beta(\mathscr{P})$ to be the protocol $(D', V', Q', q_0', M', \mu', \delta', \gamma')$, which is given by the following.

- $D' = \{\text{IDLE, REQUEST}\} \times D$. An element of D' is denoted $[b, c]$ where $b \in \{\text{IDLE, REQUEST}\}$ and where $c \in D$. The first component of D' is used to simulate requests to protocol $\mathscr{P}$ and the second is used to simulate the random choices made in $\mathscr{P}$.
- $V' = \{1, 2\}$.
- $Q' = Q^2$. An element of Q' is denoted $[q_1, q_2]$ where $q_1 \in Q$ and where $q_2 \in Q$.
- $q_0' = [q_0, q_0]$.
- $M' = M^2$. An element of M' is denoted $[m_1, m_2]$ where $m_1 \in M$ and where $m_2 \in M$.
- $\mu'_{p,q}$ is defined

$$\mu'_{p,q}(r, v, [b, c], [q_1, q_2]) = [\mu_{p,q}(r, w_1, c, q_1), \mu_{p,q}(r, w_2, c, q_2)],$$

where $w_i = b$ if $i = v$ and $w_i = \text{IDLE}$ otherwise. The role played by w_i is to provide requests to the one simulated copy of protocol $\mathscr{P}$ that corresponds to the input to processor p and to block requests to the other simulated copy of $\mathscr{P}$.

- δ'_p is defined as

$$\delta'_p([m_1, m_1'], ..., [m_n, m_n']) = [\delta_p(m_1, ..., m_n), \delta_p(m_1', ..., m_n')],$$

where, in the domain of δ'_p, we identify NULL with [NULL, NULL].

- γ'_p is defined as

$$\gamma'_p([q_1, q_2]) = \min\{i \mid \gamma_p(q_i) = \text{FIRE}\},$$

where we follow the convention that min $\varnothing$ = UNDECIDED.

We define two functions, first and second. For any lazy distributed firing squad protocol $\mathscr{P}$, these functions yield executions of $\mathscr{P}$ when given an execution of the protocol $\beta(\mathscr{P})$. In Lemma 6 we will use these functions to characterize the relationship between executions of the protocol $\mathscr{P}$ and executions of the protocol $\beta(\mathscr{P})$. Let $\mathscr{P} = (D, Q, q_0, M, \mu, \delta, \gamma)$ be any protocol that solves the lazy distributed firing squad problem, let $\mathscr{P}' = \beta(\mathscr{P})$, and let $E' = (C', \langle i_1, ..., i_n \rangle, H)$ be any execution of protocol $\mathscr{P}'$. Recall that C' is a function from $N \times \mathscr{I}^+$ to $\{\text{IDLE, REQUEST}\} \times D$. Let $B: N \times \mathscr{I}^+ \to \{\text{IDLE, REQUEST}\}$ and $C: N \times \mathscr{I}^+ \to D$ be chosen so that C' is the cross product of the two functions B and C. Thus $C'(p, r) = [B(p, r), C(p, r)]$ for all $(p, r) \in N \times \mathscr{I}^+$. For all $j \in \{1, 2\}$ let W_j be the request history

$$W_j(p, r) = \begin{cases} B(p, r) & \text{if } i_p = j; \\ \text{IDLE} & \text{otherwise.} \end{cases}$$

We define $\text{first}(E') = (C, W_1, H)$ and $\text{second}(E') = (C, W_2, H)$.

LEMMA 6. *Let $\mathscr{P} = (D, Q, q_0, M, \mu, \delta, \gamma)$ be any protocol that solves the lazy distributed firing squad problem, let $\mathscr{P}' = \beta(\mathscr{P})$, and let $E' = (C', \langle i_1, ..., i_n \rangle, H)$ be any execution of protocol $\mathscr{P}'$. If $E_1 = \text{first}(E')$ and $E_2 = \text{second}(E')$, then for all $(p, r) \in N \times \{0, 1, ...\}$*

$$\text{state}(p, r, E') = \begin{cases} [\text{state}(p, r, E_1), \text{state}(p, r, E_2)] & \text{if } r = 0 \text{ or } H(p, r, n) = \text{SENT}; \\ \text{CRASHED} & \text{otherwise.} \end{cases}$$

Proof. Suppose that $\mathscr{P}' = (D', V', Q', q'_0, M', \mu', \delta', \gamma')$. The proof is by induction on r.

Basis ($r = 0$): We calculate that

$$\begin{aligned} \text{state}(p, 0, E') &= [q_0, q_0] \\ &= [\text{state}(p, 0, E_1), \text{state}(p, 0, E_2)]. \end{aligned}$$

Induction: If processor p crashes by round r in message history H then it is immediate that $\text{state}(p, r, E') = \text{CRASHED}$. So, for the remainder of the proof we suppose that processor p does not crash by round r. Thus $H(p, r, n) = \text{SENT}$.

For all $(q, s, j) \in N^2 \times \{1, 2\}$, let

$$m_{q,s} = \begin{cases} \mu'_{q,s}(r, i_q, C'(q, r), \text{state}(q, r-1, E')) & \text{if } H(q, r, s) = \text{SENT}; \\ \text{NULL} & \text{if } H(q, r, s) = \text{SILENT}, \end{cases}$$

and let

$$m^j_{q,s} = \begin{cases} \mu_{q,s}(r, W_j(q, r), C(q, r), \text{state}(q, r-1, E_j)) & \text{if } H(q, r, s) = \text{SENT}; \\ \text{NULL} & \text{if } H(q, r, s) = \text{SILENT}. \end{cases}$$

We claim that $m_{q,s} = [m^1_{q,s}, m^2_{q,s}]$ for all $(q, s) \in N^2$. If $H(q, r, s) = \text{SILENT}$ then it is immediate that the claim is true. If $H(q, r, s) = \text{SENT}$ then (using Lemma 6 and the definitions of $m_{q,s}$, $m^1_{q,s}$, $m^2_{q,s}$, B, and C) we calculate that

$$\begin{aligned} m_{q,s} &= \mu'_{q,s}(r, i_q, C'(q, r), \text{state}(q, r-1, E')) \\ &= \mu'_{q,s}(r, i_q, [B(q, r), C(q, r)], [\text{state}(q, r-1, E_1), \text{state}(q, r-1, E_2)]) \\ &= [\mu_{q,s}(r, W_1(q, r), C(q, r), \text{state}(q, r-1, E_1)), \\ &\quad \mu_{q,s}(r, W_2(q, r), C(q, r), \text{state}(q, r-1, E_2))] \\ &= [m^1_{q,s}, m^2_{q,s}]. \end{aligned}$$

Having proved the claim that $m_{q,s} = [m^1_{q,s}, m^2_{q,s}]$ for all $(q, s) \in N^2$, we now conclude the induction step by calculating that

$$\begin{aligned} \text{state}(p, r, E') &= \delta'_p(m_{1,p}, \ldots, m_{n,p}) \\ &= \delta'_p([m^1_{1,p}, m^2_{1,p}], \ldots, [m^1_{n,p}, m^2_{n,p}]) \\ &= [\delta_p(m^1_{1,p}, \ldots, m^1_{n,p}), \delta_p(m^2_{1,p}, \ldots, m^2_{n,p})] \\ &= [\text{state}(p, r, E_1), \text{state}(p, r, E_2)]. \quad \blacksquare \end{aligned}$$

THEOREM 7. *Let $\mathscr{P} = (D, Q, q_0, M, \mu, \delta, \gamma)$ be any protocol that solves the lazy distributed firing squad problem. If $\mathscr{P}' = \beta(\mathscr{P})$, then the protocol $\mathscr{P}'$ solves the lazy simultaneous weak agreement problem.*

Proof. Suppose that $\mathscr{P}' = (D', V', Q', q'_0, M', \mu', \delta', \gamma')$. We show that the agreement, simultaneity, and validity conditions are satisfied.

Agreement and simultaneity conditions: Say that correct processor p decides v in round r of execution E' of protocol $\mathscr{P}'$. Let $E_1 = \text{first}(E')$ and let $E_2 = \text{second}(E')$. By Lemma 6, $\text{state}(q, r, E') = [\text{state}(q, r, E_1), \text{state}(q, r, E_2)]$ for all correct processors q. It is immediate from the definition of decides that $\gamma'_p(\text{state}(p, r, E')) = v$ and that

$\gamma'_p(\text{state}(p, r', E')) = \text{UNDECIDED}$ for all $r' \in \{1, ..., r-1\}$. Using the definition of γ', we observe that

$$\gamma_p(\text{state}(p, r, E_v)) = \text{FIRE},$$

$$\gamma_p(\text{state}(p, r', E_j)) = \text{UNDECIDED for all } (r', j) \in \{1, ..., r-1\} \times \{1, 2\},$$

and

$$\text{if } v = 2 \text{ then } \gamma_p(\text{state}(p, r, E_1)) = \text{UNDECIDED}.$$

By the simultaneity condition satisfied by protocol $\mathscr{P}$ we have, for all correct processors q

$$\gamma_q(\text{state}(q, r, E_v)) = \text{FIRE},$$

$$\gamma_q(\text{state}(q, r', E_j)) = \text{UNDECIDED for all } (r', j) \in \{1, ..., r-1\} \times \{1, 2\},$$

and

$$\text{if } v = 2 \text{ then } \gamma_q(\text{state}(q, r, E_1)) = \text{UNDECIDED}.$$

Let q be any processor that is correct in execution E'. Using the definition of γ', we have that $\gamma'_q(\text{state}(q, r, E')) = v$ and that $\gamma'_q(\text{state}(q, r', E')) = \text{UNDECIDED}$ for all $r' \in \{1, ..., r-1\}$. Thus any correct processor decides v in round r of execution E'.

Validity condition: Say that all processors are correct and that all processors start execution E' of protocol $\mathscr{P}'$ with input v. There are two cases. Either $v = 1$ or $v = 2$. We argue the case when $v = 1$. The other case is similar. Let $E_1 = \text{first}(E')$ and let $E_2 = \text{second}(E')$. By Lemma 6, $\text{state}(p, r, E') = [\text{state}(p, r, E_1), \text{state}(p, r, E_2)]$ for all $p \in N$. By the definition of the function second and by the choice of input in the execution E', no processor ever receives a request to fire in execution E_2. Thus $\gamma_p(\text{state}(p, r, E_2)) = \text{UNDECIDED}$ for all $(p, r) \in N \times \mathscr{I}^+$. From the definition of γ' we have that $\gamma'_p(\text{state}(p, r, E')) \in \{\text{UNDECIDED}, 1\}$ for all $(p, r) \in N \times \mathscr{I}^+$. Thus the decision of all of the processors that decide is 1. ∎

LEMMA 8. *Let $\mathscr{P}$ be any protocol that solves the lazy distributed firing squad problem. Let $E = (C, W, H)$ be any normal execution of protocol $\mathscr{P}$ in which the correct processors fire in some round r. If $\mathscr{P}' = \beta(\mathscr{P})$ then there is some normal execution E' of protocol $\mathscr{P}'$ in which the correct processors decide by round r.*

Proof. This proof is in two parts. First we construct the execution E'. Second we show that it has the desired properties.

Construction of the execution E': We specify that $E' = (C', \langle 1, 1, ..., 1 \rangle, H)$ where the random-choice history C' is defined to be $C'(p, r') = [W(p, r'), C(p, r')]$ for all $(p, r') \in N \times \mathcal{I}^+$.

Verification that the execution E' *has the desired properties*: We now show that execution E' is normal and that all correct processors decide by round r in execution E'.

Executions E and E' have the same message histories. Execution E is a normal execution of a lazy distributed firing squad protocol. The following property follows from the definitions of normal executions of lazy distributed firing squad and randomized simultaneous agreement protocols: if H' is the message history of any normal execution of a lazy distributed firing squad protocol then any execution of a randomized simultaneous agreement protocol with message history H' is normal. Thus execution E' is normal.

Having shown that execution E' is normal, we now show that all correct processors decide by round r in execution E'. For all $p \in N$ let γ_p be the decision function used by processor p in protocol $\mathcal{P}$ and let γ'_p be the decision function used by processor p in protocol $\mathcal{P}'$. Note that $E = \text{first}(E')$. Let $F = \text{second}(E')$. By Lemma 6, $\text{state}(p, r, E') = [\text{state}(p, r, E), \text{state}(p, r, F)]$. By assumption, the correct processors fire in round r in execution E. Thus, $\gamma_p(\text{state}(p, r, E)) = \text{FIRE}$. We have that $\gamma'_p[(\text{state}(p, r, E), \text{state}(p, r, F)]) = 1$, by the definition of γ'. Therefore, the correct processors decide by round r in execution E' of protocol $\mathcal{P}'$. ∎

LEMMA 9. *If there is a protocol* $\mathcal{P}$ *that solves the lazy distributed firing squad problem and that has an* l*-normal execution* E *in which all correct processors fire in round* r*, then there is a protocol* $\mathcal{P}'$ *that solves the lazy distributed firing squad problem and that has a* 0*-normal execution* E' *in which all correct processors fire in round* $r - l$.

Proof. The protocol $\mathcal{P}'$ is only slightly different from the protocol $\mathcal{P}$; it is constructed from $\mathcal{P}$ by having each processor p start in the state $\text{state}(p, l, E)$ rather than in the sate q_0. (We overcome the technical obstacle that our model requires that all n processors start in the same state by encoding the new start states in the message generation functions of protocol $\mathcal{P}'$.) For all r, in protocol $\mathcal{P}'$ each processor sends the messages that it would send in round $r + l$ of protocol $\mathcal{P}$. That is, if protocol $\mathcal{P}$ uses the message generation function $\mu_{p,q}(r, w, c, s)$, then protocol $\mathcal{P}'$ uses the message generation function $\mu'_{p,q}(r, w, c, s) = \mu_{p,q}(r + l, w, c, s)$. No other change is required.

The proof that protocol $\mathcal{P}'$ solves the lazy distributed firing squad problem is straightforward and is omitted. The execution E' is constructed from the execution E simply by discarding the first l rounds of the random-

choice history, the request history, and the message history. Removing l quiescent rounds from the start of an l-normal execution in this way produces a 0-normal execution. Thus execution E' is 0-normal. It is straightforward to show by induction on r that $\text{state}(p, r', E') = \text{state}(p, l + r', E)$ for all $(p, r') \in N \times \mathcal{I}^+$. No processors fire in quiescent rounds of execution E by the validity condition satisfied by protocol $\mathcal{P}$. Thus, if the correct processors fire in round r in execution E then they fire in round $r - l$ in execution E'. ∎

THEOREM 10. *Let $\mathcal{P}$ be any protocol that solves the lazy distributed firing squad problem. If E is any l-normal execution of protocol $\mathcal{P}$, then the earliest round in which the correct processors can fire in execution E is round $l + t + 1$.*

Proof. The proof is by contradiction. Suppose that the correct processors fire before round $l + t + 1$ in execution E of protocol $\mathcal{P}$. By Lemma 9, there is a protocol $\mathcal{P}'$ that solves the lazy distributed firing squad problem and that has a 0-normal execution E' in which the correct processors fire before round $t + 1$.

Let $\mathcal{P}'' = \beta(\mathcal{P}')$. Protocol $\mathcal{P}''$ solves the lazy simultaneous weak agreement problem by Theorem 7. By Lemma 8, there is some normal execution of protocol $\mathcal{P}''$ in which the correct processors decide before round $t + 1$. By Theorem 5, such an execution is impossible, contradiction. ∎

It is an immediate corollary of Theorem 10 that $t + 1$ is a lower bound on the expected number of rounds required to solve the lazy distributed firing squad problem.

4. TABULATION OF KNOWN BOUNDS

We tabulate the known upper and lower bounds on rounds for various consensus problems. The variables used in the tables are n, the number of processors; t, a bound on the number of faults; and f, the number of faults that actually occur. We consider the distributed firing squad problem, the simultaneous agreement problem, and the eventual agreement problem. For each problem we consider a worst-case bound, a worst-case bound parameterized by f, and a bound on the expected number of rounds required by a randomized protocol. The lower bounds are in the crash fault model; the upper bounds are in the Byzantine fault model. For the agreement problems, the lower bounds assume weak agreement and the upper bounds assume strong agreement.

Lower bounds are given in Table 1. These bounds are for the crash fault model and therefore also hold for more malicious failure models. Each

TABLE 1

Lower Bounds for Crash Faults

	Distributed firing squad problem	Simultaneous agreement problem	Eventual agreement problem
Worst-case rounds (deterministic)	$t+1$ Coan, Dolev, Dwork, and Stockmeyer (1989)	$t+1$ Lamport and Fischer (1982)	$t+1$ Lamport and Fischer (1982)
Worst-case rounds parameterized by f (deterministic)	$t+1$ New in this paper.	$t+1$ Dolev, Reischuk, and Strong (1990)	$\min(f+2, t+1)$ Dolev, Reischuk, and Strong (1990)
Expected rounds (randomized)	$t+1$ New in this paper.	$t+1$ New in this paper.	— No non-trivial bound is known.

entry in the table gives a bound and a reference to the paper where the bound first appeared.

Upper bounds (protocols) are given in Table 2 for comparison with the lower bounds. The upper bounds are all for the unauthenticated Byzantine fault model and therefore also work in more benign fault models. In all but two cases, the bounds are tight. In the case of early-stopping eventual agreement protocols, Moses and Waarts (1988) have a protocol that achieves the lower bound if the number of processors is at least $6t+1$. The tabulated protocol works for all $n \geq 3t+1$. In the case of randomized eventual agreement protocols, there are many protocols that improve on the

TABLE 2

Upper Bounds for Unauthenticated Byzantine Faults

	Distributed firing squad problem	Simultaneous agreement problem	Eventual agreement problem
Worst-case rounds (deterministic)	$t+1$ Burns and Lynch (1987)	$t+1$ Lamport, Shostak, and Pease (1982)	$t+1$ Lamport, Shostak, and Pease (1982)
Worst-case rounds parameterized by f (deterministic)	$t+1$ Burns and Lynch (1987)	$t+1$ Lamport, Shostak, and Pease (1982)	$\min(2f+5, 2t+3)$ Dolev, Reischuk, and Strong (1990)
Expected rounds (randomized)	$t+1$ Burns and Lynch (1987)	$t+1$ Lamport, Shostak, and Pease (1982)	$O(t/\log n)$ Chor and Coan (1985)

tabulated one for more benign fault models. Protocols exist that achieve $O(1)$ rounds if n is $\Omega(t^2)$ (Ben-Or, 1985), if only crash faults occur (Chor, Merritt, and Shmoys, 1989), or if there is a trusted dealer (Rabin, 1983). There is a protocol due to Bracha (1987) that uses cryptography to terminate in $O(\log n)$ rounds. The performance of this protocol was improved to $O(\log \log n)$ rounds by Dwork, Shmoys, and Stockmeyer (1990). Another protocol by Dwork, Shmoys, and Stockmeyer uses cryptography to terminate in $O(1)$ rounds if n is $\Omega(t \cdot \log t)$.

The protocol of Lamport, Shostak, and Pease (1982) is a deterministic simultaneous agreement protocol that always terminates in $t+1$ rounds. It is therefore also a degenerate randomized protocol (in which processors ignore their random choices) that terminates in an optimal expected number of rounds. Each entry in the table gives a bound and a reference to the paper where the bound first appeared.

APPENDIX A: Lower Bounds for Deterministic Agreement

We prove Theorem 1, which is a lower bound on the number of rounds required by any deterministic protocol for the lazy simultaneous weak agreement problem. All executions and protocols in this appendix are deterministic. Our proof is in the style developed by Merritt (1984) as simplified by Dwork and Moses (1990).

A.1. *Directly Similar Executions and Similar Executions*

We define two useful relations on executions—direct similarity and similarity. Recall that t is an upper bound on the number of faulty processors. Let E and E' be executions of some protocol. E is *directly similar* to E', written $E \sim E'$, if there is some processor p such that $\text{state}(p, t, E) = \text{state}(p, t, E')$, and p is correct in E and E'. Note the special role played by round t: two executions are directly similar if they are indistinguishable to some correct processor at round t. The relation $\sim$ is symmetric and reflexive. The relation *similar*, written $\approx$, is taken to be the transitive closure of the relation $\sim$. Thus, $\approx$ is an equivalence relation.

We now prove three lemmas that establish some basic properties of the $\sim$ relation. In Lemma A1 we prove that two executions are directly similar if they differ only in the sending of a single round t message (i.e., the message is sent in one execution and is not sent in the other).

Lemma A1. *Let $E = (\mathbf{0}, I, H)$ and $E' = (\mathbf{0}, I, H')$ be arbitrary executions. Let p and q be arbitrary processors. If the message histories H and H' are identical except that $H(p, t, q) = \text{SENT}$ and $H'(p, t, q) = \text{SILENT}$, then $E \sim E'$.*

Proof. We claim that state(s, r, E) = state(s, r, E') for all $(s, r) \in N \times \{0, ..., t\} - \{(p, t), (q, t)\}$. The proof of the claim is a straightforward induction on r and is omitted.

Processor p is faulty in execution E'. In every execution there are at least two correct processors because $n \geq t + 2$. So, there must be some processor $u \neq q$ that is correct in execution E'. Any processor that is correct in execution E' is also correct in execution E. Thus processor u is correct in executions E and E'. By the claim, state(u, t, E) = state(u, t, E'). Thus $E \sim E'$. ∎

In Lemma A2 we prove that two executions are directly similar if, for some r, they differ only in the sending of a single round r message to some processor that crashes (in both executions) before sending any round $r + 1$ messages.

LEMMA A2. *Let $E = (\mathbf{0}, I, H)$ and $E' = (\mathbf{0}, I, H')$ be arbitrary executions. Let $r \in \mathscr{I}^+$. Let p and q be arbitrary processors. If $H(q, r + 1, 1) =$ SILENT and if the message histories H and H' are identical except that $H(p, r, q) =$ SENT and $H'(p, r, q) =$ SILENT, then $E \sim E'$.*

Proof. We claim that state(s, r', E) = state(s, r', E') for all $(s, r') \in N \times \{0, ...\} - \{(p, r), (q, r)\}$. The proof of the claim is a straightforward induction on r' and is omitted.

Processors p and q are faulty in execution E'. In every execution there is at least one correct processor because $n \geq t + 2$. So there must be some processor u that is correct in execution E'. Any processor that is correct in execution E' is also correct in execution E. Thus processor u is correct in executions E and E'. By the claim, state(u, t, E) = state(u, t, E'). Thus $E \sim E'$. ∎

In Lemma A3 we prove that two executions are directly similar if they differ only in the input to some processor that crashes before sending its first message.

LEMMA A3. *Let $E = (\mathbf{0}, I, H)$ and $E' = (\mathbf{0}, I', H)$ be arbitrary executions, where $I = \langle i_1, ..., i_n \rangle$ and where $I' = \langle i'_1, ..., i'_n \rangle$. Let p be an arbitrary processor. If $i_q = i'_q$ for all $q \in N - \{p\}$ and if $H(p, 1, 1) =$ SILENT, then $E \sim E'$.*

Proof. We claim that state(s, r, E) = state(s, r, E') for all $(s, r) \in N \times \{0, ...\} - (p, 0)$. The proof of the claim is a straightforward induction on r and is omitted.

Clearly, executions E and E' have the same set of faulty processors. In every execution there is at least one correct processor because $n \geq t + 2$. So, there must be some processor u that is correct in executions E and E'. By the claim, state(u, t, E) = state(u, t, E'). Thus $E \sim E'$. ∎

A.2. *The Lower Bound*

In this subsection we give a series of lemmas that culminate in our lower bound for agreement. Lemma A4 is our key lemma in which we prove that two normal executions are similar if they differ only in the sending of a single round $r \in \{1, ..., t\}$ message (i.e., the message is sent in one execution and not in the other) and if no processor ever crashes after sending its last round r message.

LEMMA A4. *Let* $E = (\mathbf{0}, I, H)$ *and* $E' = (\mathbf{0}, I, H')$ *be arbitrary normal executions. Let* $r \in \{1, ..., t\}$. *Let* p *and* q *be arbitrary processors. If every processor that is faulty in message history* H *crashes by round* r *and if the message histories* H *and* H' *are identical except that* $H(p, r, q) = \text{SENT}$ *and* $H'(p, r, q) = \text{SILENT}$, *then* $E \approx E'$.

Proof. The proof is by reverse induction on r, from $r = t$ to $r = 1$.

Basis ($r = t$): By Lemma A1, $E \sim E'$. Thus $E \approx E'$.

Induction: We make the following definitions. Let

$$J(s, r', u) = \begin{cases} \text{SILENT} & \text{if } r' \geqslant r+1 \text{ and } s = q; \\ H(s, r', u) & \text{otherwise.} \end{cases}$$

The message history J is identical to the message history H except that in message history J processor q crashes after sending its last round r message. Let

$$J'(s, r', u) = \begin{cases} \text{SILENT} & \text{if } r' \geqslant r+1 \text{ and } s = q; \\ H'(s, r', u) & \text{otherwise.} \end{cases}$$

The message history J' is identical to the message history H' except that in message history J' processor q crashes after sending its last round r message. Let $F = (\mathbf{0}, I, J)$ and let $F' = (\mathbf{0}, I, J')$. Observe that F and F' are normal executions. (Both the fact that F and F' are executions and the fact that they are normal are deduced from the following: E and E' are normal executions and $r + 1 \leqslant t$.) Observe that every processor that is faulty in message history J crashes by round $r + 1$ and every processor that is faulty in message history J' crashes by round $r + 1$. We now calculate

$E \approx F$ By repeated application of the induction hypothesis.

$\sim F'$ By Lemma A2.

$\approx E'$ By repeated application of the induction hypothesis. ∎

In Lemma A5 we use Lemmas A3 and A4 to show that two failure-free

normal executions are similar if they differ only in the input to one processor.

LEMMA A5. *Let* $E = (\mathbf{0}, I, H)$ *and* $E' = (\mathbf{0}, I', H)$ *be arbitrary normal executions where* $I = \langle i_1, ..., i_n \rangle$ *and where* $I' = \langle i'_1, ..., i'_n \rangle$. *Let* p *be an arbitrary processor. If* $i_q = i'_q$ *for all* $q \in N - \{p\}$ *and if all processors are correct in the message history* H, *then* $E \approx E'$.

Proof. We make the following definition. Let

$$J(s, r', u) = \begin{cases} \text{SILENT} & \text{if } s = p; \\ \text{SENT} & \text{otherwise.} \end{cases}$$

Thus, the message history J is identical to the message history H except that in message history J processor p sends no messages. Let $F = (\mathbf{0}, I, J)$ and let $F' = (\mathbf{0}, I', J)$. We calculate that

$E \approx F$ By repeated application of Lemma A4.

$\sim F'$ By Lemma A3.

$\approx E'$ By repeated application of Lemma A4. ∎

In Lemma A6 we use Lemmas A4 and A5 to prove the surprising result that all normal executions are similar.

LEMMA A6. *For all normal executions* $E = (\mathbf{0}, I, H)$ *and* $E' = (\mathbf{0}, I', H')$ *it is the case that* $E \approx E'$.

Proof. Let $J(p, r, q) = \text{SENT}$ for all $r \in \mathcal{I}^+$ and for all $(p, q) \in N^2$. Let $F = (\mathbf{0}, I, J)$ and let $F' = (\mathbf{0}, I', J)$. We calculate that

$E \approx F$ By repeated application of Lemma A4.

$\approx F'$ By repeated application of Lemma A5.

$\approx E'$ By repeated application of Lemma A4. ∎

THEOREM 1. *Let* $\mathcal{P}$ *be any deterministic protocol that solves the lazy simultaneous weak agreement problem. In any normal execution of* $\mathcal{P}$ *no correct processor decides before round* $t + 1$.

Proof. The proof is by contradiction. Suppose there is a normal execution $E = (\mathbf{0}, I, H)$ of protocol $\mathcal{P}$ in which some correct processor decides in some round r where $r \leqslant t$. By the simultaneity condition satisfied by $\mathcal{P}$, all correct processors decide in round r. By the agreement condition satisfied by $\mathcal{P}$, all correct processors reach the same decision. Without loss of generality suppose they all decide 0.

Let $H'(s, r', u) = \text{SENT}$ for all $r' \in \mathscr{I}^+$ and for all $(s, u) \in N^2$. Let $E' = (\mathbf{0}, I', H')$, where I' is an n-element vector of ones. By the validity condition satisfied by protocol $\mathscr{P}$, no correct processor decides 0 in execution E'. By Lemma A6, $E \approx E'$. Thus there is a chain of executions $E = E_1 \sim E_2 \sim \cdots \sim E_j = E'$. We have already shown that all of the correct processors in execution $E = E_1$ decide 0 in round r. For all $i \in \{2, ..., j\}$, it follows from the definition of $\sim$ and from the agreement and simultaneity conditions satisfied by protocol $\mathscr{P}$ that all of the correct processors in execution E_i decide 0 in round r. Thus all of the correct processors in E' decide 0 in round r, contradiction. ∎

ACKNOWLEDGMENTS

We thank Michael Merritt for sharing with us his deep understanding of the $(t+1)$-round lower bound for deterministic Byzantine agreement. We also thank Yoram Moses, Nancy Lynch, and the referees for their helpful comments on earlier versions of this paper and Larry Stockmeyer for a helpful conversation.

RECEIVED January 12, 1988; FINAL MANUSCRIPT RECEIVED November 17, 1989

REFERENCES

BEN-OR, M. (1985), Fast asynchronous Byzantine agreement, *in* "Proceedings, 4th ACM Symposium on Principles of Distributed Computing," pp. 149–151.

BRACHA, G. (1987), An $O(\log n)$ expected rounds randomized Byzantine generals protocol, *J. Assoc. Comput. Mach.* **34**, 910–920.

BURNS, J., AND LYNCH, N. A. (1987), The Byzantine firing squad problem, *in* "Advances in Computing Research: Parallel and Distributed Computing," Vol. 4, JAI Press, Greenwich, CT.

CHOR, B., AND COAN, B. A. (1985), A simple and efficient randomized Byzantine agreement algorithm, *IEEE Trans. Software Engrg.* **SE-11**, 531–539.

CHOR, B., MERRITT, M., AND SHMOYS, D. (1989), Simple constant-time consensus protocols in realistic failure models, *J. Assoc. Comput. Mach.* **36**, 591–614.

COAN, B. A., DOLEV, D., DWORK, C., AND STOCKMEYER, L. (1989), The distributed firing squad problem, *SIAM J. Comput.* **18**, 990–1012.

COAN, B. A., AND DWORK, C. (1986), Simultaneity is harder than agreement, *in* "Proceedings, 5th IEEE Symposium on Reliability in Distributed Software and Database Systems," pp. 141–150.

DEMILLO, R., LYNCH, N. A., AND MERRITT, M. (1982), Cryptographic protocols, *in* "Proceedings, 14th ACM Symposium on Theory of Computing," pp. 383–400.

DOLEV, D., REISCHUK, R., AND STRONG, H. R. (1990), Early stopping in Byzantine agreement, *J. Assoc. Comput. Mach.* **37**, 720–741.

DOLEV, D., AND STRONG, H. R. (1982), Polynomial algorithms for multiple processor agreement, *in* "Proceedings, 14th ACM Symposium on Theory of Computing," pp. 401–407.

DWORK, C., AND MOSES, Y. (1990), Knowledge and common knowledge in a Byzantine environment: Crash failures, *Inform. and Comput.* **88**, 156–186.

DWORK, C., SHMOYS, D., AND STOCKMEYER, L. (1990), Flipping persuasively in constant time, *SIAM J. Comput.* **19**, 472–499.

FISCHER, M. J., AND LYNCH, N. A. (1982), A lower bound for the time assure interactive consistency, *Inform. Process. Lett.* **14**, 183–186.

HADZILACOS, V. (1984), "Issues of Fault Tolerance in Concurrent Computations," Ph.D. Thesis, Harvard University. (Available as Technical Report TR-11-84, Department of Computer Science, Harvard University, 1984.)

HALPERN, J. Y., AND MOSES, Y. (1990), Knowledge and common knowledge in a distributed environment, *J. Assoc. Comput. Mach.* **37**, 549–587.

LAMPORT, L., AND FISCHER, M. J. (1982), Byzantine generals and transaction commitment protocols, unpublished manuscript.

LAMPORT, L., SHOSTAK, R. E., AND PEASE, M. (1982), The Byzantine generals problem, *ACM Trans. Programming Languages Syst.* **4**, 382–401.

MERRITT, M. (1983), "Cryptographic Protocols," Ph.D. Thesis, Georgia Institute of Technology.

MERRITT, M. (1984), private communication.

MOSES, Y., AND TUTTLE, M. R. (1988), Programming simultaneous actions using common knowledge, *Algorithmica* **3**, 121–169.

MOSES, Y., AND WAARTS, O. (1988), Coordinated traversal: $(t+1)$-Round Byzantine agreement in polynomial time, *in* "Proceedings, 29th IEEE Symposium on Foundations of Computer Science," pp. 246–255.

RABIN, M. O. (1983), Randomized Byzantine generals, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science," pp. 403–409.