Reaching Approximate Agreement in the Presence of Faults

Danny Dolev[1]

Nancy A. Lynch[2]

Shlomit S. Pinter[3]

Eugene W. Stark[4]

William E. Weihl[5]

October 2, 1985

Abstract

This paper considers a variant on the Byzantine Generals problem, in which processes start with arbitrary real values rather than Boolean values or values from some bounded range, and in which approximate, rather than exact, agreement is the desired goal. Algorithms are presented to reach approximate agreement in asynchronous, as well as synchronous systems. The asynchronous agreement algorithm is an interesting contrast to a result of Fischer, Lynch, and Paterson, who show that exact agreement is not attainable in an asynchronous system with as few as one faulty process. The algorithms work by successive approximation, with a provable convergence rate that depends on the ratio between

[1] Hebrew University, Jerusalem, Israel

[2] Massachusetts Institute of Technology, Cambridge, MA. This work was supported in part by the NSF under Grant No. DCR-8302391, U.S. Army Research Office Contract #DAAG29-84-K-0058, and the Defense Advanced Research Projects Agency (DARPA) #N00014-83-K-0125.

[3] Technion, Haifa, Israel

[4] State University of New York at Stony Brook, Stony Brook, NY

[5] Massachusetts Institute of Technology, Cambridge, MA. This work was supported in part by a graduate fellowship from the Fannie and John Hertz Foundation, and the Defense Advanced Research Projects Agency (DARPA) #N00014-83-K-0125.

the number of faulty processes and the total number of processes. Lower bounds on the convergence rate for algorithms of this form are proven, and the algorithms presented are shown to be optimal.

1 Introduction

In designing fault-tolerant distributed systems, one often encounters questions of agreement among processes. In the Byzantine Generals problem [PSL 80, LSP 82], the objective is for nonfaulty processes to agree on a value, in spite of the presence of a small number of "Byzantine" types of faults — completely arbitrary, even possibly malicious, behavior. Several variations on the problem can be considered — the model can be synchronous or asynchronous, and either exact or approximate agreement can be demanded. In this paper, we consider a variant on the traditional Byzantine Generals problem, in which processes start with arbitrary real values, and where approximate, rather than exact, agreement is the desired goal. Approximate agreement can be used, for example, for clock synchronization and for stabilization of input from sensors.

We assume a model in which processes can send messages containing arbitrary real values, and can store arbitrary real values as well. We assume that each process starts with an arbitrary real value. For any preassigned $\epsilon > 0$ (as small as desired), an *approximate agreement algorithm* must satisfy the following two conditions:

- Agreement: All nonfaulty processes eventually halt with output values that are within ϵ of each other.

- Validity: The value output by each nonfaulty process must be in the range of initial values of the nonfaulty processes.

Thus, in particular, if all nonfaulty processes should happen to start with the same initial value, the final values are all required to be the same as the common initial value. This is consistent with the usual requirements for Byzantine agreement algorithms. However, should the nonfaulty processes start with different values, we do not require that the nonfaulty processes agree on a unique final value.

We consider both synchronous and asynchronous versions of the problem. Systems in which there is a finite bounded delay on the operations of the processes and on their intercommunication are said to be synchronous. In such systems, unannounced process deaths, as well as long delays, are considered to be faults. For synchronous systems, we give a simple and rather efficient algorithm for achieving approximate agreement. This algorithm works by successive approximation, with a provable convergence rate that depends on the ratio between the number of faulty processes and the total number of processes. The algorithm is guaranteed to converge when the total number of processes is more than three times the number of possible faulty processes. Termination is achieved using a technique that ensures that all nonfaulty processes halt, but allows different processes to terminate at different times.

For asynchronous systems, in which a very slow process cannot be distinguished from a dead process, no exact agreement can be achieved, even if no malicious failures occur [FLP 83, DDS 83]. An interesting contrast to the results in [FLP 83, DDS 83] is our second algorithm, which enables processes in an asynchronous system to get as close to agreement as one chooses. Our algorithm for the asynchronous case also works by successive approximation. In this case, however, the total number of processes required by the algorithm is more than five times the number of possible faulty processes. As in the synchronous case, we achieve termination using a technique that ensures that all nonfaulty processes halt, but permits different processes to terminate at different times.

Our algorithms to obtain approximate agreement are of a very simple form. Namely, at each round until termination is reached, each process sends its latest value to all processes (including itself). On receipt of a collection V of values, the process computes a certain function $f(V)$ as its next value. The function f is a kind of averaging function. Here we use functions that are appropriate for handling t faulty processes. We will show that these functions have particularly nice approximation behavior. In particular, we will show that, for algorithms of a particular form, no approximation function can provide uniformly faster convergence than the functions used in this paper. An earlier paper [DLPSW 83] presented similar algorithms, but used approximation functions that provided slower convergence than achieved by the functions used in this paper.

The remainder of this paper is organized as follows: In Section 2, we prove some combinatorial properties of the approximation functions upon which our algorithms depend. Then, in Section 3, we introduce the synchronous model and present the synchronous approximate agreement algorithm, and in Section 4, we present the asynchronous model and algorithm. Next, in Section 5, we present lower bounds on the convergence rate for algorithms of the form presented in sections 3 and 4, and show that the approximation functions used in our algorithms are optimal. In Section 6, we discuss the resilience properties of our algorithms. Finally, in Section 7, we conclude with a short summary and some open questions.

2 Properties of the Approximation Functions

In this section, we will state and prove the relevant properties of the approximation functions. First, we require some preliminary definitions and properties of multisets.

2.1 Preliminary Definitions

Let $\mathcal{N}$ be the natural numbers, including 0, and let $\mathcal{R}$ be the real numbers. We view a finite multiset U of reals as a function $U : \mathcal{R} \rightarrow \mathcal{N}$ that is nonzero on at most finitely many $r \in \mathcal{R}$. Intuitively, the function U assigns a finite multiplicity to each value $r \in \mathcal{R}$. The *cardinality* of a multiset U is given by $\sum_{r \in \mathcal{R}} U(r)$, and is denoted by $|U|$. We say that a multiset is *empty* if its cardinality is zero; otherwise it is *nonempty*. The *difference* $U - V$ of multisets U and V is the multiset W defined by

$$W(r) = \begin{cases} U(r) - V(r) & \text{if } U(r) - V(r) \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

The *intersection* $U \cap V$ of multisets U and V is the multiset W defined by $W(r) = \min(U(r), V(r))$.

In the sequel, the term "multiset" will always refer to finite multisets of real numbers as above. If g is a function on multisets, then g^k will denote the k-fold iteration of g; thus $g^1 = g$, $g^2 = g \circ g$, etc.

The *minimum* $\min(U)$ of a nonempty multiset U is defined by

$$\min(U) = \min\{r \in \mathcal{R} : U(r) \neq 0\}.$$

The *maximum* $\max(U)$ is defined similarly. If U is nonempty, let $\rho(U)$ (the *range* of U) be the interval $[\min(U), \max(U)]$, and let $\delta(U)$ (the *diameter* of U) be $\max(U) - \min(U)$. The *mean* $\text{mean}(U)$ of a nonempty multiset U is defined by

$$\text{mean}(U) = \sum_{r \in \mathcal{R}} rU(r)/|U|.$$

If U is a nonempty multiset, we define the multiset $s(U)$ (intuitively, the multiset obtained by removing one occurrence of the smallest value in U) to be the multiset W defined by

$$W(r) = \begin{cases} U(r) - 1 & \text{if } \ r = \min(U), \\ U(r) & \text{otherwise.} \end{cases}$$

The multiset $l(U)$ (remove one occurrence of the largest value in U) is defined similarly. If $|U| \geq 2$, then define $\text{reduce}(U) = s(l(U))$, the result of removing the largest and smallest elements of U.

The first lemma shows that the number of common elements in two nonempty multisets is reduced by at most 1 when the smallest (or the largest) element is removed from each.

Lemma 1 *Suppose that V and W are nonempty multisets. Then*

1. $|V \cap W| - |s(V) \cap s(W)| \leq 1$.
2. $|V \cap W| - |l(V) \cap l(W)| \leq 1$.

Proof – We prove the first inequality; the argument for the second is symmetric. If V and W have the same minimum, then the same element is removed from each, and hence at most one element is removed from their intersection. If the minima of V and W are not the same, then either the minimum of V is not in W, or the minimum of W is not in V. In either case, at most one element is removed from the intersection. ∎

The next lemma extends the results of the previous lemma to removing the j largest and j smallest elements.

Lemma 2 *Suppose that j is a nonnegative integer and that V and W are multisets such that $|V| \geq 2j$ and $|W| \geq 2j$. Then*

$$|V \cap W| - |\text{reduce}^j(V) \cap \text{reduce}^j(W)| \leq 2j.$$

Proof – Follows from repeated application of Lemma 1. ∎

The next lemma is fundamental to the correctness of the algorithms. It states that if V and U are multisets such that V contains at most j values not in U, then every value in $\text{reduce}^j(V)$ is in the range of U. For example, if the multiset of values held by nonfaulty processes at some point in the algorithm is U, and the multiset of values received by some process is V, then at most t of the values in V are not in U, where t is the maximum number of faulty processes. The lemma then states that $\text{reduce}^t(V)$ is a multiset whose range is contained in the range of the values of the nonfaulty processes. This property is essential in showing that the validity condition is satisfied.

Lemma 3 *Suppose that j is a nonnegative integer and that U and V are nonempty multisets such that $|V - U| \le j$ and $|V| > 2j$. Then $\rho(\text{reduce}^j(V)) \subseteq \rho(U)$.*

Proof – Suppose $\rho(\text{reduce}^j(V)) \not\subseteq \rho(U)$. Then either $\min(\text{reduce}^j(V)) < \min(U)$ or $\max(\text{reduce}^j(V)) > \max(U)$. If $\min(\text{reduce}^j(V)) < \min(U)$, then $\sum_{r<\min(U)} V(r) \ge j+1$. Hence, $|V - U| \ge j+1$, which contradicts a hypothesis. The case $\max(\text{reduce}^j(V)) > \max(U)$ is symmetric. ∎

2.2 The Approximation Functions

Suppose U is a nonempty multiset. Let $m = |U|$ and let $u_0 \le u_1 \le \ldots \le u_{m-1}$ be the elements of U in nondecreasing order. If $k > 0$ then define $\text{select}_k(U)$ to be the multiset consisting of the elements $u_0, u_k, u_{2k}, \ldots$, and u_{jk}, where $j = \lfloor (m-1)/k \rfloor$. Thus, $\text{select}_k(U)$ chooses the smallest element of U and every kth element thereafter.

An important role will be played by the constants

$$c(m,k) = \lfloor (m-1)/k \rfloor + 1,$$

where $c(m,k)$ is the number of elements in $\text{select}_k(U)$ when U has m elements. The constant $c(n-2t,t)$ appears as the convergence factor for the synchronous protocol, and the constant $c(n-3t,2t)$ as the convergence factor for the asynchronous protocol.

In this paper we will use approximation functions drawn from a class of functions parametrized by: (1) the number t of faulty processes, and (2) a constant k, the choice of which depends on t and on whether the algorithm is synchronous or asynchronous. For $k > 0$ and $t \geq 0$ define the function $f_{k,t}$ by

$$f_{k,t}(V) = \text{mean}(\text{select}_k(\text{reduce}^t(V))),$$

for all multisets V with $|V| > 2t$. The approximation function for the synchronous protocol with no more than t faulty processes is $f_{t,t}$. The approximation function for the asynchronous protocol with no more than t faulty processes is $f_{2t,t}$. We will show below why these functions are appropriate.

The next two lemmas describe properties of the approximation functions. Lemma 4 is used in verifying the validity condition.

Lemma 4 *Suppose $k > 0$ and $t \geq 0$ are integers. Suppose that U and V are nonempty multisets such that $|V - U| \leq t$ and $|V| > 2t$. Then $f_{k,t}(V) \in \rho(U)$.*

Proof – Follows easily from Lemma 3 (with $j = t$). ∎

Lemma 5 will be applied to determine the rate of convergence of the approximation rounds. The multisets V and W will be the multisets of values received by two nonfaulty processes in a given round, and U will be the multiset of values held by nonfaulty processes at the beginning of that round. Nonfaulty processes use the appropriate approximation function to choose their values for the next round; the lemma tells us how quickly those values converge.

Lemma 5 *Suppose V, W, and U are multisets, and $k > 0$, $t \geq 0$, and $m > 2t$ are integers, with $|V| = |W| = m$, $|V - U| \leq t$, $|W - U| \leq t$, and $|W - V| = |V - W| \leq k$. Then*

$$|f_{k,t}(V) - f_{k,t}(W)| \leq \delta(U)/c(m - 2t, k).$$

Proof – Let $M = \text{reduce}^t(V)$ and $N = \text{reduce}^t(W)$. Since V and W each contain exactly m elements, M and N each contain exactly $m - 2t$ elements, and hence $\text{select}_k(M)$ and $\text{select}_k(N)$ each contain exactly $c = c(m - 2t, k)$ elements. Let $m_0 \leq m_1 \leq \ldots \leq m_{c-1}$ be the elements of $\text{select}_k(M)$, and let $n_0 \leq n_1 \leq \ldots \leq n_{c-1}$ be the elements of $\text{select}_k(N)$. Notice that there are at least $ki + 1$ elements in M that are less than or equal to m_i, and at most ki elements in M that are strictly less than m_i; similarly for N.

We begin by showing that $\max(m_i, n_i) \leq \min(m_{i+1}, n_{i+1})$ for $0 \leq i \leq c-2$. It suffices to show that $m_i \leq n_{i+1}$; a symmetric argument demonstrates that $n_i \leq m_{i+1}$.

We proceed by contradiction: Suppose that $m_i > n_{i+1}$. As noted above, there are at least $k(i+1)+1$ elements in N less than or equal to n_{i+1}. By our supposition, these elements are strictly less than m_i. However, there are at most ki elements in M strictly less than m_i. Therefore, there are at least $k(i+1)+1-ki$ $(=k+1)$ elements in N that are not in M; thus, $|N-M| \geq k+1$. Now by hypothesis, $|W-V| \leq k$, so $|W \cap V| \geq m-k$. Then Lemma 2 shows $|N \cap M| \geq m-k-2t$, and hence $|N-M| \leq (m-2t)-(m-k-2t) = k$. This is a contradiction, and we conclude that $m_i \leq n_{i+1}$.

Now we will use the inequality shown above to obtain the desired result. Using the notation defined above,

$$
\begin{aligned}
|f_{k,t}(V) - f_{k,t}(W)| &= |\mathrm{mean}(\mathrm{select}_k(M)) - \mathrm{mean}(\mathrm{select}_k(N))| \\
&= |(\sum_{i=0}^{c-1} m_i) - (\sum_{i=0}^{c-1} n_i)|/c \\
&= |\sum_{i=0}^{c-1}(m_i - n_i)|/c \\
&\leq \sum_{i=0}^{c-1}|m_i - n_i|/c \qquad \text{(by the triangle inequality)} \\
&= \sum_{i=0}^{c-1}(\max(m_i, n_i) - \min(m_i, n_i))/c.
\end{aligned}
$$

By the inequality demonstrated above, for $0 \leq i \leq c-2$, $(\max(m_i, n_i) - \min(m_i, n_i)) \leq (\min(m_{i+1}, n_{i+1}) - \min(m_i, n_i))$, so we get

$$
\begin{aligned}
|f_{k,t}(V) - f_{k,t}(W)| \leq\ & [\max(m_{c-1}, n_{c-1}) - \min(m_{c-1}, n_{c-1})]/c \\
& + \sum_{i=0}^{c-2}(\min(m_{i+1}, n_{i+1}) - \min(m_i, n_i))/c.
\end{aligned}
$$

Collecting terms then shows that

$$|f_{k,t}(V) - f_{k,t}(W)| \leq (\max(m_{c-1}, n_{c-1}) - \min(m_0, n_0))/c.$$

Now, $\rho(M) \subseteq \rho(U)$ and $\rho(N) \subseteq \rho(U)$ by Lemma 3 (with $j = t$), so $\max(m_{c-1}, n_{c-1}) \leq \max(U)$ and $\min(m_0, n_0) \geq \min(U)$. Hence

$$|f_{k,t}(V) - f_{k,t}(W)| \leq (\max(U) - \min(U))/c$$

$$= \delta(U)/c,$$

as desired. ∎

3 The Synchronous Problem

A *synchronous approximation algorithm* P is a system of n processes, $n \geq 1$. Each process p has a set of states, including a subset of states called *initial states* and a subset called *halting states*. There is a *value* mapping which assigns a real number as the value of each state. For each real number r, there is exactly one initial state with value r. Each process acts deterministically according to a *transition function* and a *message generation function*. The transition function takes a non-halting process state and a vector of messages received from all processes (one message per process) and produces a new process state. The message generation function takes a non-halting state and produces a vector of messages to be sent to all processes (one per process).

We assume that the system acts synchronously, using a reliable communication medium. Each process is able to send messages to all processes (including itself), and the sender of each message is identifiable by the receiver.

A *configuration* consists of a state for each process. An *initial configuration* consists of an initial state for each process. Let T be any subset of the processes. A sequence of configurations (called *rounds*), $C_0, C_1, C_2, \ldots$ is a T*-computation* provided there exist messages sent by each process at each round such that: (a) C_0 is an initial configuration; (b) for every i, and every $p \in T$, the messages sent out by p after C_i are exactly those specified by p's message generation function, applied to p's state in C_i; and (c) for every i, and every $p \in T$, p's state in C_{i+1} is exactly the one specified by p's transition function applied to p's state in C_i and the messages sent to p after C_i. In a T-computation, processes in T are nonfaulty, while processes not in T may be faulty.

For the rest of the paper, assume a fixed small value ϵ, a fixed number of processes n, and a fixed maximum number of faulty processes t.

A synchronous approximation algorithm is said to be *t-correct* provided that for every subset T of processes with $|T| \geq n - t$, and every T-computation, the following is true:

Every $p \in T$ eventually enters a halting state, and the following two conditions hold for the values of those halting states:

- Agreement: If two processes in T enter halting states with values r and s, respectively, then $|r - s| \leq \epsilon$.
- Validity: If a process in T enters a halting state with value r, then there exist processes in T having x and y as initial values, such that $x \leq r \leq y$.

We will prove the following theorem.

Theorem 1 *If $n \geq 3t+1$, then there exists a t-correct synchronous approximation algorithm with n processes.*

Note that the following strategy would suffice to prove Theorem 1. The processes could run n executions of a general (unlimited value set) Byzantine Generals algorithm such as the one in [DS 82], in order to obtain common estimates for the initial values of all the processes. After this algorithm completes, all processes in T will have the same multiset, V, of values for all the processes. Then each process halts with value $f(V)$, where f is a predetermined averaging function that is the same for all processes. This algorithm actually achieves exact real-valued agreement, with the required validity condition. However, the solution presented below is simpler and more elegant, and moreover extends directly to the asynchronous case, for which exact agreement is impossible. The algorithm has two additional advantages over using a Byzantine Generals algorithm: it is more resilient than typical Byzantine Generals algorithms, and it can, in some cases, terminate in fewer than $t + 1$ rounds.

We now present our synchronous approximation algorithm, S. First, we describe a non-terminating algorithm, S_0, and then we discuss how termination is achieved. We assume that $n \geq 3t + 1$.

Synchronous Approximation Algorithm S_0:

At each round, each nonfaulty process p performs the following steps:

1. Process p broadcasts its current value to all processes, including itself.

2. Process p collects all the values sent to it at that round into a multiset V. If p does not receive exactly one correct value from some particular other process (which means, in the synchronous model, that the other process is faulty), then p simply picks some arbitrary default value to represent that process in the multiset. The multiset V will therefore always contain exactly n values.

3. Process p applies the function $f_{t,t}$ to the multiset V to obtain its new value.

The following result states how the diameter and range of the nonfaulty processes' values are affected by each round of algorithm S_0.

Lemma 6 *Suppose $n, t > 0$ are such that $n \geq 3t + 1$. Let T be a set of processes, with $|T| \geq n - t$. Let h be a positive integer. Let U and U' be the multisets of values of processes in T, immediately before and after round h, respectively, in a particular T-computation of S_0. Then*

1. $\delta(U') \leq \delta(U)/c(n-2t, t)$.

2. $\rho(U') \subseteq \rho(U)$.

Proof – Let p and q be arbitrary processes in T. Let V and W be the multisets of values (including default values) received by p and q, respectively, at round h. Then $|V| = |W| = n$. Since there are at most t faulty processes, $|V - U| \leq t$ and $|W - U| \leq t$. Moreover, since V and W contain identical entries for all the processes in T, we know that $|V - W| = |W - V| \leq t$.

1. The multisets V, W, and U satisfy the hypotheses of Lemma 5 (with $m = n$ and $k = t$). Thus,

$$|f_{t,t}(V) - f_{t,t}(W)| \leq \delta(U)/c(n-2t, t).$$

2. The multisets V and U satisfy the hypotheses of Lemma 4. Thus $f_{t,t}(V) \in \rho(U)$.

Since p and q were chosen arbitrarily, the result follows. ∎

Part 1 of Lemma 6 shows that, at each round, the diameter of the multiset of values held by nonfaulty processes decreases by a factor of $c(n-2t, t)$, which is at least 2 because

$n \geq 3t + 1$. Thus, the diameter of the multiset of values held by nonfaulty processes eventually decreases to ϵ or less. In addition, repeated application of part 2 of Lemma 6 shows that, at each round $h \geq 1$, the values held by nonfaulty processes immediately before round h are all in the range of the initial values of nonfaulty processes.

It is now easy to see why the function $f_{t,t}$ is appropriate for the synchronous algorithm. Since a correct process can receive at most t values in a round from faulty processes, t-fold application of reduce is sufficient to ensure that extreme values from faulty processes are discarded. Thus, the second subscript of f is t. Also, if p and q are correct processes that receive multisets V and W, respectively, in a round, then t is the maximum number of values that can be in $V - W$. Application of select_t to the reduced multisets is therefore sufficient to obtain convergence, and the first subscript of f is also t.

Algorithm S_0 is not a correct synchronous approximation algorithm, however, for as stated, it never terminates. We modify S_0 to obtain a terminating algorithm, S, as follows. At the first round, each nonfaulty process uses the range of all the values it has received at that round to compute a round number at which it is sure that the values of any two nonfaulty processes will be at most ϵ apart. Each process can do this because it knows the value of ϵ, the guaranteed rate of convergence and furthermore, it knows that the range of values it receives on the first round includes the initial values of all nonfaulty processes. The total number of rounds that must be executed (including the first round) is given by $\lceil \log_c(\delta(V)/\epsilon) \rceil$, where V is the multiset of values received in the first round, and $c = c(n - 2t, t)$.

In general, different processes might compute different round numbers. Any process that reaches its computed round simply halts, and sends its value out with a special "halting" tag. When any process, say p, receives a value with a halting tag, it knows to use the enclosed value not only for the designated round, but also for all future rounds (until p itself decides to halt, based on p's own computed round number). Although nonfaulty processes might compute different round numbers, it is clear that the smallest such estimate is correct. Thus, at the time the first nonfaulty process halts, the range is already sufficiently small. At subsequent rounds, the range of values of nonfaulty processes is never increased, although we can no longer guarantee that it decreases. The following lemma makes these ideas more precise.

Lemma 7 *Assume that $n \geq 3t+1$. Let T be a set of processes, with $|T| \geq n-t$. Let h be a positive integer. Let U and U' be the multisets of values of processes in T, immediately before and after round h, respectively, in a particular T-computation of S. Then $\rho(U') \subseteq \rho(U)$.*

Proof – Let p be an arbitrary process in T. Let v and v' be the values held by p immediately before and after round h, respectively. It suffices, since p is arbitrary, to show that $v' \in \rho(U)$. If p has terminated prior to the start of round h, then $v' = v \in \rho(U)$. If p has not halted prior to the start of round h, then let V be the multiset of values received by p in round h. Then V and U satisfy the hypotheses of Lemma 4, and since $v' = f_{t,t}(V)$, it follows that $v' \in \rho(U)$. ∎

Algorithm S is summarized in Figure 1. To show that S is a correct synchronous approximation algorithm, we must show that all processes terminate, and that the agreement and validity conditions are satisfied. It is clear that all processes terminate. Consider the agreement property. At the first round at which some nonfaulty process halts, it is already the case that all nonfaulty processes' values are within ϵ of each other. By Lemma 7, this diameter never increases at subsequent rounds, so the final values of all the nonfaulty processes are also within ϵ of each other. The validity property also follows from repeated application of Lemma 7. This completes the proof of Theorem 1.

As a final note, observe that algorithm S can be modified so that a process need not always wait for its computed round to arrive before halting: it can halt after it receives halting tags from at least $t+1$ other processes.

4 The Asynchronous Problem

In this section, we reformulate the problem in an asynchronous model adapted from the one in [FLP 83]. In an *asynchronous approximation algorithm*, we assume that processes have states as before, but now the operation of the processes is described by a transition function that in one step tries to receive a message, gets back either "null" or an actual message, and based on the message, changes state and sends out a finite number of other

Figure 1: Synchronous Approximation Algorithm S

Round 1 (First Approximation Round):

Input v;

$V \leftarrow \mathit{SynchExchange}(v)$;

$v \leftarrow f_{t,t}(V)$;

$H \leftarrow \lceil \log_c(\delta(V)/\epsilon) \rceil, \qquad$ where $c = c(n-2t, t)$.

Round h $(2 \leq h \leq H)$ (Approximation Rounds):

$V \leftarrow \mathit{SynchExchange}(v)$;

$v \leftarrow f_{t,t}(V)$.

Round $H+1$ (Termination Round):

$\mathit{Broadcast}(\langle v, \mathbf{halted} \rangle)$;

Output v.

Subroutine *SynchExchange*(v):

Broadcast(v)

Collect n responses:

- Fill in values for halted processes.
- Fill in default values, if necessary.

Return the multiset of responses.

messages. Nonfaulty processes always follow the algorithm. Faulty processes, on the other hand, are constrained so that their steps at least follow the standard form — in each step, they try to receive a message as do nonfaulty processes. However, they can change state arbitrarily (not necessarily according to the given algorithm), and can send out any finite set of messages (not necessarily the ones specified by the algorithm). A *T-computation* of an asynchronous approximation algorithm is one in which the processes in T always follow the algorithm, all processes (faulty and nonfaulty) continue to take steps until they reach a halting state, and any process that fails to enter a halting state eventually receives all messages sent to it.

An asynchronous approximation algorithm is said to be *t-correct* provided for every subset T of processes with $|T| \geq n - t$, and every T-computation, every process in T eventually halts, and the same agreement and validity conditions hold as for the synchronous case.

It seems simplest here to insist on the standard form being followed by all processes. The requirement that faulty processes keep taking steps until they enter halting states is not a restriction, since they are free to enter halting states at any time they wish. Similarly, the requirement that faulty processes continue trying to receive messages is not a restriction, since they are free to do whatever they like with the messages received. Finally, the requirement that faulty processes only send finitely many messages at each step is needed so that faulty processes are unable to flood the message system, preventing messages from other processes from getting through.

We assume that processes take steps at completely arbitrary rates, so that there is no way (in finite time) to distinguish a faulty process from one that is simply slow in responding. Also, we assume that the message system takes arbitrary lengths of time to deliver messages, and delivers them in arbitrary order.

We will prove the following theorem:

Theorem 2 *If $n \geq 5t + 1$, then there exists a t-correct asynchronous approximation algorithm with n processes.*

We now describe the asynchronous approximation algorithm. As in the synchronous case, first we describe a nonterminating algorithm, A_0, in which processes compute better and better approximations, and we then modify A_0 to produce a terminating algorithm A. Assume that $n \geq 5t+1$.

Asynchronous Approximation Algorithm A_0

At round h, each nonfaulty process p performs the following steps:

1. Process p labels its current value with the current round number h, and then broadcasts this labeled value to all processes, including itself.

2. Process p waits to receive exactly $n-t$ round h values, and collects these values into a multiset V. Since there can be at most t faulty processes, process p will eventually receive at least $n-t$ round h values. Note that, in contrast to the synchronous case, process p does *not* choose any default values.

3. Process p applies the function $f_{2t,t}$ to the multiset V to obtain its new value.

In analogy with Lemma 6, we have the following result, which states the convergence properties of the above algorithm.

Lemma 8 *Suppose $n, t > 0$ are such that $n \geq 5t+1$. Let T be a set of processes, with $|T| \geq n-t$. Let h be a positive integer. Let U and U' be the multisets of values of processes in T, immediately before and after round h, respectively, in a particular T-computation of A_0. Then*

1. $\delta(U') \leq \delta(U)/c(n-3t, 2t)$.

2. $\rho(U') \subseteq \rho(U)$.

Proof – Let p and q be arbitrary processes in T. Let V and W be the multisets of values received by p and q, respectively, at round h. Then $|V| = |W| = n-t$. Since there are at most t faulty processes, $|V-U| \leq t$ and $|W-U| \leq t$. Moreover, since V and W both contain identical entries for all the processes in T from which both p and q heard, we know that $|V \cap W| \geq n-3t$. Hence $|V-W| = |W-V| = |V| - |V \cap W| \leq 2t$.

1. The multisets V, W, and U satisfy the hypotheses of Lemma 5 (with $m = n-t$ and $k = 2t$). Thus,

$$|f_{2t,t}(V) - f_{2t,t}(W)| \leq \delta(U)/c(n-3t, 2t).$$

2. The multisets V and U satisfy the hypotheses of Lemma 4. Thus $f_{2t,t}(V) \in \rho(U)$.

Since p and q were chosen arbitrarily, the result follows. ∎

Part 1 of Lemma 8 shows that, at each round, the diameter of the multiset of values of nonfaulty processes decreases by a factor of $c(n-3t, 2t)$, which is at least 2 because $n \geq 5t+1$. Thus, the diameter of the multiset of values held by nonfaulty processes eventually decreases to ϵ or less. In addition, repeated application of part 2 of Lemma 8 shows that, at each round $h \geq 1$, the values held by nonfaulty processes immediately before round h are all in the range of the initial values of nonfaulty processes.

We can now see why $f_{2t,t}$ is the appropriate approximation function for the asynchronous algorithm. The second subscript is t because, as in the synchronous case, that is the maximum number of values a correct process can receive in a round that are not values of correct processes. The first subscript is $2t$ because if the correct processes p and q receive multisets V and W, respectively, in a round, then $2t$ is the maximum number of values that can be in $V - W$ (t faulty values, plus t nonfaulty values received by p but not by q).

The only remaining problem is termination. We cannot use the same technique that we used in the synchronous algorithm, because a process cannot wait until it hears from all other processes, and thus cannot obtain an estimate of the range of the initial values of the nonfaulty processes. We solve this problem by adding an initialization round at the beginning of the algorithm. In this initialization round (round 0), each nonfaulty process p performs the following steps:

Initialization Round for Asynchronous Approximation Algorithm A:

1. Process p labels its initial value with the round number 0, and then broadcasts this labeled value to all processes, including itself.

2. Process p waits to receive exactly $n-t$ round 0 values, and collects these values into a multiset V_p.

3. Process p chooses an arbitrary element of $\rho(\text{reduce}^{2t}(V_p))$ (say $\text{mean}(\text{reduce}^{2t}(V_p))$) as its initial value for use in round 1. Let x_p be this chosen value.

Suppose that p and q are arbitrary nonfaulty processes. Then since $|V_p| > 4t$ and $|V_p - V_q| \leq 2t$, it follows that V_p and V_q satisfy the hypotheses for the multisets V and U, respectively, in Lemma 3 (with $j = 2t$). An application of this result therefore shows that, for any nonfaulty processes p and q, it is the case that $x_p \in \rho(V_q)$. That is, the value x_p computed by process p as the result of the initialization round is contained in the range of all values received by process q in the initialization round. Since each nonfaulty process q knows: (1) that its range $\rho(V_q)$ contains all the round 1 values x_p for nonfaulty processes p; (2) the value of ϵ; and (3) the guaranteed rate of convergence, it can compute, before the beginning of round 1, a round number at which it is sure that the values of any two nonfaulty processes will be at most ϵ apart. The total number of rounds that must be executed by a process, not including the initialization round, is $\lceil log_c(\delta(V)/\epsilon)\rceil$, where V is the multiset received in the initialization round, and $c = c(n-3t, 2t)$.

As in the synchronous case, different processes will calculate different round numbers at which they would like to halt. The same modification, of sending a value out with a special halting tag, works here as well. We obtain a lemma analogous to Lemma 7:

Lemma 9 *Assume that $n \geq 5t+1$. Let T be a set of processes, with $|T| \geq n-t$. Let h be a positive integer. Let U and U' be the multisets of values of processes in T, immediately before and after round h, respectively, in a particular T-computation of A. Then $\rho(U') \subseteq \rho(U)$.*

Algorithm A is summarized in Figure 2. The remainder of the proof of Theorem 2 is analogous to that of Theorem 1.

5 Lower Bound Results

In this section, we assume that algorithms are of a standard form in which at each round, an old approximation is exchanged with other processes, and a new approximation is computed from the multiset of values received, by the application of an approximation function f. We assume that f is *cautious*, as defined below. (Our algorithms all fit this pattern.) The results show that, under these assumptions, the function $f_{t,t}$ gives the best possible single-round convergence factor for a synchronous algorithm for $n \geq 3t+1$, and the function $f_{2t,t}$ gives the best possible single-round convergence factor for an asynchronous algorithm for $n \geq 5t+1$.

Figure 2: Asynchronous Approximation Algorithm A

Round 0 (Initialization Round):

Input v;

$V \leftarrow AsynchExchange(v, 0)$;

$v \leftarrow \text{mean}(\text{reduce}^{2t}(V))$;

$H \leftarrow \lceil \log_c(\delta(V)/\epsilon) \rceil, \qquad \text{where } c = c(n-3t, 2t)$.

Round h $(1 \leq h \leq H)$ (Approximation Rounds):

$V \leftarrow AsynchExchange(v, h)$;

$v \leftarrow f_{2t,t}(V)$.

Round $H+1$ (Termination Round):

$Broadcast(\langle v, \textbf{halted} \rangle)$;

Output v.

Subroutine $AsynchExchange(v, h)$:

$Broadcast(\langle v, h \rangle)$

Collect $n-t$ round h responses:

- Fill in values for halted processes.
- Do not fill in default values.

Return the multiset of responses.

We should note that the results of this section merely show the existence, given a particular choice of approximation functions, of multisets that demonstrate the worst-case behavior of those approximation functions. These multisets satisfy cardinality constraints such that they *could* be the multisets appearing in some round of an actual execution of the algorithm, for example the first round. However, the multisets of values appearing in any round of an execution of the algorithm in general depend upon the behavior of the faulty processes at all preceding rounds. We do not necessarily know that the faulty processes can conspire to produce worst-case behavior at *each* round of the algorithm. The results of this section therefore do not preclude the existence of approximation functions whose per-round convergence factor is not constant over the course of the algorithm, but rather becomes more favorable as the algorithm progresses.

In [DLPSW 83], an earlier version of this work, we used different approximation functions in our algorithms. The discovery of the lower bounds in this section suggested that those functions did not give optimal rates of convergence, and led us to search for the improved approximation functions that appear in this paper.

In the remainder of this section, let n and t be fixed.

We say that an approximation function f, which takes a multiset M of real numbers to a real number $f(M)$, is *cautious* if $f(M) \in \rho(U)$ for all multisets U such that $|M-U| \leq t$. The cautious requirement seems reasonable for any approximation function that will tolerate up to t faults: regardless of the values received from the faulty processes, a cautious function will produce a value in the range of the values held by the nonfaulty processes. It is easy to see that $f_{k,t}$ is cautious for all $k > 0$.

5.1 The Synchronous Problem

We will show the following theorem:

Theorem 3 *Suppose $n, t > 0$ are such that $n \geq 3t+1$. Suppose that f and g are cautious* *approximation functions. Then there exist multisets V, W, and U such that:*

$$\begin{aligned}|V| = |W| &= n,\\ |U| &= n-t,\\ |V-U| = |W-U| &= t, \text{ and}\\ |f(V)-g(W)| &\geq \delta(U)/c(n-2t,t).\end{aligned}$$

The implications of this result for the synchronous agreement algorithm are the following: Suppose we consider algorithms of a standard form, in which at each round, a process exchanges its current approximation with all other processes, and then applies a cautious approximation function to the multiset of values it receives to determine its new approximation. Theorem 3 then implies that there exist multisets V, W, and U, such that if correct processes p and q (using approximation functions f and g, respectively) receive multisets of values V and W, respectively, in some round of execution, and U is the multiset of values held by correct processes at the start of that round, then the new approximations held by p and q at the end of the round can be no closer than $\delta(U)/c(n-2t,t)$. Thus this result yields a fundamental limitation on the rate of convergence of algorithms of the standard form. The lower bound given by this result also matches the upper bound provided by the function $f_{t,t}$.

The proof of Theorem 3 requires the following lemma, which asserts the existence of a chain of multisets that spans from a multiset M_0 upon which every cautious approximation function must yield 0, to a multiset M_c upon which every cautious approximation function must yield 1, where $c = c(n-2t,t)$. The chain is defined so that:

1. M_0 has the value 0 with multiplicity $n-t$ and the value 1 with multiplicity t, and

2. For $0 \leq i \leq c-1$, the multiset M_{i+1} is obtained from M_i by changing t of the values from 0 to 1.

Lemma 10 *Suppose $n,t > 0$ are such that $n \geq 3t+1$. Let $c = c(n-2t,t)$. Then there* *exist multisets $M_0, M_1, \ldots, M_c$, and $U_1, U_2, \ldots, U_c$ such that:*

$$\begin{aligned} |M_i| &= n \text{ for } 0 \le i \le c, \\ |U_i| &= n - t \text{ for } 1 \le i \le c, \\ |M_i - U_{i+1}| = |M_{i+1} - U_{i+1}| &= t \text{ for } 0 \le i \le c-1, \\ \delta(U_i) &= 1 \text{ for } 1 \le i \le c, \end{aligned}$$

and such that $f(M_0) = 0$ and $f(M_c) = 1$ whenever f is a cautious approximation function.

Proof – Define M_i to have the value 0 with multiplicity $n - (i+1)t$ and the value 1 with multiplicity $(i+1)t$. Define U_i to have the value 0 with multiplicity $n - (i+1)t$ and the value 1 with multiplicity it. The cardinality and diameter constraints on these sets are easily checked. Suppose f is cautious. Then since M_0 has the value 0 with multiplicity $n-t$ $(> t)$ and the value 1 with multiplicity t $(\le t)$, it follows that $f(M_0) = 0$. Also, M_c has the value 0 with multiplicity $n - (c+1)t$ and the value 1 with multiplicity $(c+1)t$. From the definition of c, we know that $n - 3t < (c-1)t + 1 \le n - 2t$, so $(c+1)t \ge n - t$, and $n - (c+1)t \le t$. It follows that $f(M_c) = 1$. ∎

We can now present the proof of Theorem 3:

Proof – For $0 \le i \le c$ $(= c(n-2t, t))$, let the approximation function h_i be f if i is even, and g if i is odd. By Lemma 10, there exists a chain $M_0, M_1, \ldots, M_c$, and $U_1, U_2, \ldots, U_c$ such that:

$$\begin{aligned} |M_i| &= n \text{ for } 0 \le i \le c, \\ |U_i| &= n - t \text{ for } 1 \le i \le c, \\ |M_i - U_{i+1}| = |M_{i+1} - U_{i+1}| &= t \text{ for } 0 \le i \le c-1, \\ \delta(U_i) &= 1 \text{ for } 1 \le i \le c, \end{aligned}$$

and such that $h_0(M_0) = 0$ and $h_c(M_c) = 1$. Suppose, to obtain a contradiction, that $|h_{i+1}(M_{i+1}) - h_i(M_i)| < 1/c$ for $0 \le i \le c-1$. Then

$$\begin{aligned} 1 &= |h_c(M_c) - h_0(M_0)| \\ &= |h_c(M_c) - h_{c-1}(M_{c-1}) + h_{c-1}(M_{c-1}) - h_{c-2}(M_{c-2}) + \ldots + h_1(M_1) - h_0(M_0)| \\ &\le |h_c(M_c) - h_{c-1}(M_{c-1})| + |h_{c-1}(M_{c-1}) - h_{c-2}(M_{c-2})| + \ldots + |h_1(M_1) - h_0(M_0)| \\ &< c/c \\ &= 1. \end{aligned}$$

This is a contradiction, and we conclude that $|h_{i+1}(M_{i+1}) - h_i(M_i)| \geq 1/c$ for some i with $1 \leq i \leq c-1$. If i is even, then $h_i = f$ and $h_{i+1} = g$, so letting $V = M_i$, $W = M_{i+1}$, and $U = U_{i+1}$ satisfies the requirements of the theorem. If i is odd, then instead let $V = M_{i+1}$, $W = M_i$, and $U = U_{i+1}$. ∎

5.2 The Asynchronous Problem

We will show the following theorem:

Theorem 4 *Suppose $n, t > 0$ are such that $n \geq 5t+1$. Suppose that f and g are cautious approximation functions. Then there exist multisets V, W, and U such that:*

$$\begin{aligned} |V| = |W| &= n - t, \\ |U| &= n - t, \\ |V - U| = |W - U| &= t, \text{ and} \\ |f(V) - g(W)| &\geq \delta(U)/c(n-3t, 2t). \end{aligned}$$

The implications of this result for the asynchronous agreement algorithm are analogous to what Theorem 3 has to say about the synchronous algorithm: there exist multisets V, W, and U, such that if correct processes p and q (using approximation functions f and g, respectively) receive multisets of values V and W, respectively, in some round of execution, and U is the multiset of values held by correct processes at the start of that round, then the new approximations held by p and q at the end of the round can be no closer than $\delta(U)/c(n-3t, 2t)$. The lower bound given by this result also matches the upper bound provided by the function $f_{2t,t}$.

As before, the theorem is proved with the aid of a chain lemma. Let $c = c(n-3t, 2t)$. The chain is defined so that:

1. M_0 has the value 0 with multiplicity $n-2t$ and the value 1 with multiplicity t, and

2. For $0 \leq i \leq c-2$, the multiset M_{i+1} is obtained from M_i by changing $2t$ of the values from 0 to 1.

3. If M_{c-1} has the value 0 with multiplicity at least $2t+1$, then M_c is obtained from M_{c-1} by changing $2t$ of the values from 0 to 1. If M_{c-1} has the value 0 with multiplicity

$\leq 2t$, then M_c is obtained from M_{c-1} by changing t of the values from 0 to 1. Note that M_{c-1} will always have the value 0 with multiplicity at least $t+1$.

Lemma 11 *Suppose $n, t > 0$ are such that $n \geq 5t+1$. Let $c = c(n-3t, 2t)$. Then there exist multisets $M_0, M_1, \ldots, M_c$, and $U_1, U_2, \ldots, U_c$, such that:*

$$\begin{aligned} |M_i| &= n-t \text{ for } 0 \leq i \leq c, \\ |U_i| &= n-t \text{ for } 1 \leq i \leq c, \\ |M_i - U_{i+1}| = |M_{i+1} - U_{i+1}| &= t \text{ for } 0 \leq i \leq c-1, \\ \delta(U_i) &= 1 \text{ for } 1 \leq i \leq c, \end{aligned}$$

and such that $f(M_0) = 0$ and $f(M_c) = 1$ whenever f is a cautious approximation function.

Proof – From the definition of c, we know that $(2c+1)t+1 \leq n \leq (2c+3)t$. We split the proof into two cases. In case $(2c+2)t+1 \leq n \leq (2c+3)t$, then define M_i to have the value 0 with multiplicity $n-(2i+2)t$ and the value 1 with multiplicity $(2i+1)t$, for each i with $0 \leq i \leq c$. Define U_i to have the value 0 with multiplicity $n-(2i+1)t$ and the value 1 with multiplicity $2it$, for each i with $1 \leq i \leq c$. In case $(2c+1)t+1 \leq n \leq (2c+2)t$, then we modify slightly the definition of M_c and U_c from the preceding case. That is, define M_c to have the value 0 with multiplicity $n-(2c+1)t$ and the value 1 with multiplicity $2ct$. Also, define U_c to have the value 0 with multiplicity $n-2ct$ and the value 1 with multiplicity $(2c-1)t$.

In both cases it is straightforward to check that the required properties hold. ∎

The proof of Theorem 4 is entirely analogous to the proof of Theorem 3.

6 Resilience

The algorithms presented in this paper have some interesting resilience properties, stronger than those usually claimed for Byzantine agreement algorithms. So far, we have only claimed that the algorithms are resilient to t different processes exhibiting Byzantine faults during the entire course of the algorithm. However, we can claim more for situations where processes fail and recover repeatedly. Our algorithms actually support resilience to any t Byzantine faulty processes at a time (under suitable definitions of faultiness at a particular time); the total number of faulty processes can be much greater than t, since we can allow different processes to be faulty at different times.

We do not give a formal presentation of our resilience properties. Rather, we just give a brief sketch of the main ideas.

First, consider the synchronous case. A faulty process is able to recover easily and reintegrate itself into the algorithm. It can reenter the algorithm at any round, just by sending an arbitrary value, collecting values and averaging them as usual to get a new value. The process also needs to obtain an estimate of the number of rounds required before termination. It can obtain such an estimate in the reentry round, just as it could in the first round.

The asynchronous case is a little more complicated. A faulty process p needs to rejoin the algorithm at some particular (asynchronous) round; however, it must be careful to rejoin at some round that is not "out of date." That is, in the absence of additional failures of p, it must be guaranteed to receive all of its messages for that and subsequent rounds. Process p could not simply wait until it received $n - t$ messages for some particular round k, since those messages might have been delivered very late, and messages for round $k + 1$ might have already been lost. However, it suffices for p to send out a "recovery" message, and await acknowledgements from $n - t$ processes carrying the number of their current round. Process p knows that the $t + 1$st smallest of these round numbers, plus 1, is an allowable round number for it to use for reentry.

The recovering process is not able to use the same method of estimating a termination round as it did initially. Therefore, it seems necessary to modify the asynchronous algorithm to enable recovering processes to obtain termination estimates when needed. An easy modification that works is to have every process piggyback its estimate of the number of rounds to termination on every message it sends. Then a recovering process can obtain a new estimate just by taking the $t + 1$st smallest of the estimates it receives at the reentry round.

7 Summary and Open Questions

We have defined a problem of approximate agreement on real numbers by processes in a distributed system. We integrated simple approximation functions into two simple-to-implement algorithms for achieving approximate agreement — one for a synchronous distributed system, and the other for an asynchronous system. In addition, we showed that both algorithms achieve the fastest possible convergence rate for algorithms of a particular form. The algorithm for an asynchronous system provides an interesting contrast to the results in [FLP 83, DDS 83], which show that exact agreement is impossible in an asynchronous system.

The ideas of this paper have been used in the design of algorithms for synchronizing clocks in distributed systems [LL 84].

For the synchronous case, it is not difficult to show that $3t+1$ processes are necessary to solve the approximate agreement problem. The proof is an adaptation of the lower bound proof in [LSP 82], and appears in [FLM 85]. For the asynchronous case, our number of processes is not optimal. In fact, it appears possible to reduce the number of processes to as few as $3t+1$. This reduction is obtained using a more complex algorithm, based on some of the interesting ideas of [B 84]. This algorithm has a slower rate of convergence than ours.

The algorithms presented here have the undesirable property that the faulty processes, by their actions in the first round, can cause the range of values received by correct processes to be arbitrarily large, and hence can cause the time to convergence to be arbitrarily long. It appears that some of the ideas of [B 84] can also be used to obtain improved initialization rounds for the algorithms that eliminate this possibility.

To obtain the lower bound results, we had to restrict our attention to algorithms of a standard form (ones that operate by broadcasting values and using received values to compute a new approximation), and to functions with a natural, but apparently restrictive property (the "cautious" property). It would be interesting to obtain answers to the following questions:

- Can the cautious property be weakened or removed entirely?
- Can algorithms not of the standard form considered here produce agreement faster?

We would also like to have a better understanding of the relationship between the number of processes and the rate of convergence for approximate agreement algorithms. For instance, the more complex asynchronous algorithm mentioned above uses fewer processes, but has a slower rate of convergence than ours. Is there a tradeoff?

We can state a variant of the approximate agreement problem which uses a fixed number, r, of rounds, and in which ϵ is not predetermined. Each process starts with a real value, as before. After r rounds, the processes must output their final values. The validity condition is the same as before. The object of the algorithm is to insure the best possible agreement, expressed as a ratio of the new diameter of the nonfaulty processes' values to the original diameter. For given n, t and r, we would like to know the best ratio.

As before, if the algorithm is constrained to operate round-by-round, applying cautious functions at each round, we obtain lower bounds which are exactly the same as are achieved by our averaging functions. However, if the algorithm is unconstrained, the best bounds we have are not at all tight. Consider the synchronous case, for example. The best upper bound we have still arises from repeated application of our averaging function $f_{t,t}$, and is approximately $(t/n)^k$. We can obtain a lower bound by extending our chain argument of this paper to a k-dimensional hypercube (along the lines in [FL 82]). This extension gives a lower bound of approximately $(t/nk)^k$. This is still a considerable gap, which we would like to see closed. Recent work of Fekete [F] has made some progress toward this goal.

Bibliography

[B 84] G. Bracha, "An Asynchronous $\lfloor (n-1)/3 \rfloor$-Resilient Consensus Protocol," Proceedings of 3rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 154-162, August 1984.

[BL 85] J. Burns and N. A. Lynch, "The Byzantine Firing Squad Problem," Submitted for publication.

[CDDS 85] B. Coan, D. Dolev, C. Dwork and L. Stockmeyer, "The Distributed Firing Squad Problem," Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, May, 1985, pp. 335-345.

[DDS 83] D. Dolev, C. Dwork and L. Stockmeyer, "On the Minimal Synchronism Needed for Distributed Consensus," Proceedings of 24th Annual Symposium on Foundations of Computer Science, pp. 393-402, Nov. 1983. (Revised version to appear in *JACM*.)

[DLPSW 83] D. Dolev, N. A. Lynch, S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching Approximate Agreement in the Presence of Faults," Proceedings of 3rd Annual IEEE Symposium on Reliability in Distributed Software and Database Systems, pp. 145-154, October 1983.

[DS 82] D. Dolev and H. R. Strong, "Polynomial Algorithms for Multiple Processor Agreement," Proceedings of the 14th ACM SIGACT Symposium on Theory of Computing, pp. 401-407, May 1982.

[F] A. Fekete, private communication.

[FL 82] N. A. Lynch and M. Fischer, "A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters 14, 4*, pp. 183-186, June 1982.

[FLM 85] M. Fischer, N. A. Lynch, and M. Merritt, "Easy Impossibility Proofs for Distributed Consensus Problems," *Distributed Computing*, Vol. 1, No. 1, 1985.

[FLP 83] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with one faulty process," *JACM* Vol. 32, No. 2 (April, 1985), pp. 374-382.

[LL 84] N. A. Lynch and J. Lundelius, "A New Fault-Tolerant Algorithm for Clock Synchronization," Proceedings of 3rd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 75-88, August 1984. (Also, revised version submitted for publication.)

[LSP 82] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. on Programming Languages and Systems*, Vol. 4, No. 2, pp. 382-401 (1982).

[PSL 80] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *JACM*, Vol. 27, No. 2, pp. 228-234 (1980).