REACHING APPROXIMATE AGREEMENT IN THE PRESENCE OF FAULTS

Danny Dolev(1)
Nancy A. Lynch(2,3)
Shlomit S. Pinter(4)
Eugene W. Stark(2,3)
William E. Weihl(2,5)

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, which shows that exact agreement is not attainable in an asynchronous system with as few as one faulty process.

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, 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 in which approximate, rather than exact, agreement is the desired goal. Approximate agreement algorithms can be used, for example, for clock synchronization and 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 $\varepsilon > 0$ (as small as desired), an *approximate agreement algorithm* must satisfy the following two conditions:

(a) Agreement: All nonfaulty processes eventually halt with output values that are within ε of each other.
(b) 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 traditional requirements for Byzantine agreement algorithms. However, should the nonfaulty processes start with different values, we do not require that the faulty 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 which depends on the ratio between the number of faults and the total number of processes. The algorithm is guaranteed to converge in the case where the total number of processes is more than three times the number of possible faults. Termination is achieved using a simple binary Byzantine agreement on whether to halt.

(1) Hebrew University Jerusalem, Israel. This work has been performed in part at Stanford University, supported in part by DARPA under Grant No. MDA903-80-C-0107.

(2) Massachusetts Institute of Technology Cambridge, Massachusetts. This work was supported in part by the Office of Naval Research Contract No. N00014-79-C-0873.

(3) This work was supported also by the NSF under Grants No. MCS-8302391 and MCS-8306854, and U.S. Army Research Office Contract No. DAAG29-79-C-0155.

(4) Boston University Boston, Massachusetts. Current address: Technion, Haifa, Israel.

(5) This work was supported also by a graduate fellowship from the Hertz Foundation.

For asynchronous systems, in which a very slow process cannot be distinguished from a dead process, no exact agreement can be achieved [FLP 83], even if the message system is synchronous and reliable and no malicious failures occur [DDS 83]. An interesting contrast to the results in [FLP 83, DDS 83] is our second approximation 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 faults. The results in [FLP 83] suggest that no binary decision can be made for terminating the algorithm. Thus, we use an alternative technique which ensures that all nonfaulty processes halt, yet different processes can 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 vector R of values, a process computes a certain function f(R) as its next value. The function f is a kind of averaging function. Here we use two particular functions which seem appropriate for handling t faults. Both functions ignore the t largest and t smallest values in R. One function computes the mean of the remaining values, and the other function computes the midpoint of the range of the remaining values. We will show that these functions have particular nice approximation behavior. The ratio between t, the number of faulty processes, and n, the total number of participating processes, determines which of the functions results in faster convergence.

In most agreement algorithms an upper bound t on the total number of processes that become faulty during the execution of the algorithm is assumed. The complexity of the algorithm and its correctness depend on this bound. The algorithms we present might require several rounds to converge, during which time processes might become faulty and recover again. Therefore it is interesting to consider a bound on the number of concurrent faults rather than on the total number of faults. Byzantine agreement algorithms having such resiliency appears in [R 83]. We will discuss the resiliency of each algorithm we present.

The reminder of this paper is organized as follows: In Section 2 we prove some combinatorial properties of approximation functions, upon which our algorithms depend. In Section 3 we introduce the synchronous model and present the synchronous approximate agreement algorithm. In Section 4, we present the asynchronous model and algorithm. In Section 5 we discuss the resiliency properties of our algorithms. In Section 6 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.

We view a finite multiset U of reals as a function $U:\mathscr{R}\rightarrow\mathscr{N}$ which is nonzero on at most finitely many $r\in\mathscr{R}$. Intuitively, the function U assigns a finite multiplicity to each value $r\in\mathscr{R}$. The *cardinality* of a multiset U is given by $\Sigma_{r\in\mathscr{R}}U(r)$ 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 $W(r) = \min(U(r),V(r))$.

The *minimum* min(U) of a nonempty multiset U is defined by $\min(U) = \min\{r\in\mathscr{R}: U(r)\neq 0\}$. The *maximum* max(U) is defined similarly. Let $\rho(U)$ be the interval $[\min(U),\max(U)]$, and let $\delta(U)=\max(U)-\min(U)$. The *mean* of the multiset U is defined by

$$\text{mean}(U) = \Sigma_{r\in\mathscr{R}}r\cdot U(r)/|U|.$$

The *midpoint* of the multiset U is defined by
$$\text{mid}(U)=[\max(U)+\min(U)]/2.$$

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

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

The multiset $\ell(U)$ (remove one occurrence of the largest value in U) is defined similarly. Assume t is a fixed nonnegative integer. If $|U|\geq 2t$, then define $\text{reduce}(U)=s^t(\ell^t(U))$.

Throughout the paper, g^k denotes the k-fold iteration of g; thus $g^1 = g$, $g^2 = g\cdot g$, etc. In the sequel, the term "multiset" will always refer to finite multisets of real numbers as above.

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

(a) $|V\cap W| - |s(V)\cap s(W)| \leq 1$, and

(b) $|V\cap W| - |\ell(V)\cap\ell(W)| \leq 1$.

Proof: We prove the first inequality; the argument for the second is symmetric. Let $M=V \cap W$, and let $N=s(V) \cap s(W)$. Let $v=\min(V)$ and $w=\min(W)$. Now, $N(r)=M(r)$ if $r \neq v$ and $r \neq w$, so we can write $|M|-|N| = \Sigma_{r \in \{v,w\}} M(r) - \Sigma_{r \in \{v,w\}} N(r)$. There are two cases, depending whether $v=w$ or $v \neq w$.

Case $v=w$: Then

$$|M| - |N| = M(v) - N(v)$$

$$= \min(V(v), W(v)) -$$

$$\min(V(v) - 1, W(v) - 1) \leq 1.$$

Case $v \neq w$: Then

$$|M| - |N| = (M(v) + M(w)) - (N(v) + N(w))$$

$$= (\min(V(v),W(v)) + \min(V(w),W(w)))$$

$$- (\min(V(v) - 1,W(v)) + \min(V(w),W(w) - 1)).$$

Assume without loss of generality that $v<w$. Then $W(v)=0$, and so

$$|M| - |N| =$$

$$\min(V(w),W(w)) - \min(V(w),W(w) - 1) \leq 1. \quad \square$$

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

Lemma 2: Suppose that V and W are multisets such that $|V| \geq 2t$ and $|W| \geq 2t$. Then

$$|V \cap W| - |\mathrm{reduce}(V) \cap \mathrm{reduce}(W)| \nleq 2t.$$

Proof: Follows from repeated application of Lemma 1. $\square$

Lemma 3: Suppose that k is a nonnegative integer and that U and V are nonempty multisets such that $|V-U| \leq kt$ and $|V| > 2kt$. Then

$$\rho(\mathrm{reduce}^k(V)) \subseteq \rho(U).$$

Proof: Suppose $\rho(\mathrm{reduce}^k(V)) \nsubseteq \rho(U)$. Then either

$$\min(\mathrm{reduce}^k(V)) < \min(U)$$

or

$$\max(\mathrm{reduce}^k(V)) > \max(U).$$

Both cases lead to a contradiction. We argue the first; the second is symmetric.

If

$$\min(\mathrm{reduce}^k(V)) < \min(U),$$

then

$$\Sigma_{r<\min(U)} V(r) \geq kt + 1.$$

Hence, $|V-U| \geq kt+1$, which contradicts a hypothesis. $\square$

Let V be a finite multiset of reals with $|V| > 2t$. The two approximation functions we use in the paper are:

midpoint function: $f_M(V) \equiv \mathrm{mid}(\mathrm{reduce}(V))$;
mean function: $f_A(V) \equiv \mathrm{mean}(\mathrm{reduce}(V))$.

The above three lemmas describe properties of multisets that are useful for both approximation functions. In the following subsections we describe specific properties of each one of them.

2.1 The Midpoint Function

The next three lemmas describe properties of f_M. Lemma 4 verifies the validity properties; Lemmas 5 and 6 are used in verifying convergence.

Lemma 4: Suppose that U and V are nonempty multisets such that $|V-U| \leq t$ and $|V| > 2t$. Then $f_M(V) \in \rho(U)$.

Proof: Follows easily from Lemma 3. $\square$

The following lemma presents the basic reason for choosing the midpoint function as an approximation function.

Lemma 5: Suppose, U, M, and N are nonempty multisets such that $|M \cap N| > 0$, $\rho(M) \subseteq \rho(U)$, and $\rho(N) \subseteq \rho(U)$. Then $|\mathrm{mid}(M) - \mathrm{mid}(N)| \leq \delta(U)/2$.

Proof: From the definition of *mid*, we see that

$$|\mathrm{mid}(M) - \mathrm{mid}(N)| =$$

$$= |\max(M) + \min(M) - \max(N) - \min(N)|/2$$

$$= |(\max(M) - \min(N)) - (\max(N) - \min(M))|/2$$

There are two cases, depending on whether the quantity inside the absolute value sign on the previous line is nonnegative or nonpositive. Without loss of generality assume it is nonnegative. Then

$$|\mathrm{mid}(M) - \mathrm{mid}(N)| =$$

$$[(\max(M) - \min(N)) - (\max(N) - \min(M))]/2$$

$$\leq [\max(U) - \min(U)]/2 = \delta(U)/2,$$

where the inequality is obtained by noting that

$$\max(M) - \min(N) \leq \max(U) - \min(U),$$

and, since $|M \cap N| > 0$, it must be the case that $\max(N) \geq \min(M)$. $\square$

Lemma 5 describes the essential properties needed for convergence: if U is the multiset of the correct processes' values, and M and N are the reduced multisets of two correct processes, then we must show that the range of U bounds M and N, and that M and N intersect in at least one point. The earlier lemmas suggest how to establish these properties.

Lemma 6: Suppose that U, V, and W are nonempty multisets such that:

$$|V - U| \le t,$$

$$|W - U| \le t, \text{ and}$$

$$|V \cap W| > 2t.$$

Then $|f_M(V)-f_M(W)| \le \delta(U)/2$.

Proof: Let M=reduce(V) and N=reduce(W). By Lemma 2, $|M\cap N| \ge |V\cap W|-2t$. By hypothesis, $|V\cap W|>2t$, so $|M\cap N|>0$. By Lemma 3 (with k=1), $\rho(M)\subseteq\rho(U)$ and $\rho(N)\subseteq\rho(U)$. Thus, the hypotheses of Lemma 5 are satisfied, and the result follows. □

2.2 The Mean Function

Analogous to the preceding approximation function, the following lemmas present the corresponding properties of the mean function.

Lemma 7: Suppose that U and V are nonempty multisets such that $|V-U| \le t$ and $|V|>2t$. Then $f_A(V) \in \rho(U)$.

Proof: Follows easily from Lemma 3. □

Lemma 8: Suppose U, M, and N are nonempty multisets and m and i are nonnegative integers such that:

$$|M| = |N| = m,$$

$$|M \cap N| \ge i,$$

$$\rho(M) \subseteq \rho(U), \text{ and}$$

$$\rho(N) \subseteq \rho(U).$$

Then $|\text{mean}(M)-\text{mean}(N)| \le \delta(U)(m-i)/m$.

Proof: Let $L=M\cap N$, and let $M'=M-L$ and $N'=N-L$. Then

$$|\text{mean}(M) - \text{mean}(N)| =$$

$$|\Sigma_{r\in\mathscr{R}} r\cdot M(r) - \Sigma_{r\in\mathscr{R}} r\cdot N(r)|/m$$

$$= |\Sigma_{r\in\mathscr{R}} r\cdot(L(r) + M'(r)) - \Sigma_{r\in\mathscr{R}} r\cdot(L(r) + N'(r))|/m$$

$$= |\Sigma_{r\in\mathscr{R}} r\cdot M'(r) - \Sigma_{r\in\mathscr{R}} r\cdot N'(r)|/m .$$

Without loss of generality assume

$$\Sigma_{r\in\mathscr{R}} r\cdot M'(r) - \Sigma_{r\in\mathscr{R}} r\cdot N'(r) \ge 0,$$

then

$$|\text{mean}(M) - \text{mean}(N)| =$$

$$(\Sigma_{r\in\mathscr{R}} r\cdot M'(r) - \Sigma_{r\in\mathscr{R}} r\cdot N'(r))/m$$

$$\le (\max(M)\Sigma_{r\in\mathscr{R}} M'(r) - \min(N)\Sigma_{r\in\mathscr{R}} N'(r))/n$$

$$= (\max(M)(|M| - |L|) - \min(N)(|N| - |L|))/m$$

$$= (\max(M) - \min(N))(m - |L|)/m$$

Now the hypotheses that $\rho(M)\subseteq\rho(U)$, $\rho(N)\subseteq\rho(U)$, and $|M \cap N| \ge i$, imply

$$|\text{mean}(m) - \text{mean}(N)|$$

$$\le \delta(U)(m - |L|)/m$$

$$\le \delta(U)(m - i)/m . \quad □$$

Lemma 9: Suppose U, V, and W are nonempty multisets and m and i are positive integers such that:

$$|V| = |W| = m + 2t,$$

$$|V\cap W| \ge i + 2t,$$

$$|V - U| \le t, \quad \text{and}$$

$$|W - U| \le t .$$

Then $|f_A(V)-f_A(W)| \le \delta(U)(m-i)/m$.

Proof: Let M=reduce(V), and let N=reduce(W). Note that $|M|=|N|=m$. By Lemma 3, $\rho(M)\subseteq\rho(U)$ and $\rho(N)\subseteq\rho(U)$. By Lemma 2, $|M\cap N| \ge |V\cap W|-2t$. By hypothesis, $|V\cap W| \ge i+2t$, so $|M\cap N| \ge i$. Thus, the hypotheses of Lemma 8 are satisfied, and the result follows. □

3. THE SYNCHRONOUS PROBLEM

An *approximation protocol* P is a system of n processes, $n\ge1$. 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 assigned 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 it is assumed that 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 configuration (called *rounds*), C_0, C_1, C_2, ... is a *T-computation* provided there exist messages 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.

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

An approximation protocol 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 $p \in T$ enters a halting state at the same round, and the following two conditions hold for the values of those halting states.

(a) Agreement: If two processes in T enter halting states with values r and r', respectively, then $|r-r'| \leq \varepsilon$.
(b) Validity: If a process in T enters a halting state with value r, then there exist processes in T having r' and r'' as initial values, such that $r' \leq r \leq r''$.

We will prove the following theorem.

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

Before we present our algorithm, note that the following strategy suffices for obtaining Theorem 1. Namely, 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 either f_A or f_M, but is the same for all the processes. This algorithm actually achieves exact real-valued agreement, with the required validity condition. However, the solution presented below seems more elegant, and moreover extends directly to the asynchronous case, for which exact agreement is impossible.

We now present our synchronous approximation protocol. Assume that $n \geq 3t+1$. Let the function f be either f_A or f_M. These two functions have different convergence properties, which will be noted below. First, we describe a nonterminating algorithm, then we discuss how termination is achieved. At each round, each nonfaulty process p performs the following steps:

Synchronous Approximation Algorithm:

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 a single 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 to the multiset V to obtain its new value.

The following lemma states the convergence properties of the above algorithm.

Lemma 10: Assume that $n \geq 3t+1$. Let P be a synchronous approximation protocol in which each process uses the Synchronous Approximation Algorithm above. Suppose that T is a set of processes, with $|T| \geq n-t$. Let C be a T-computation of P and let k be a round number. Let U be the multiset of values held by processes in T immediately before round k in C, and let U' be the multiset of values held by processes in T immediately after round k in C. Then

(a) If $f = f_M$, then $\delta(U') \leq \delta(U)/2$.

(b) If $f = f_A$, then $\delta(U') \leq t\delta(U)/(n-2t)$.

(c) If either $f = f_A$ or $f = f_M$, then $\rho(U')$ is contained in $\rho(U)$.

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 k. 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 $|V \cap W| \geq n-t > 2t$.

(a) U, V, and W satisfy the hypotheses of Lemma 6, which therefore shows that $|f_M(V)-f_M(W)| \leq \delta(U)/2$. Since p and q were chosen arbitrarily, the result follows.

(b) Let m=n-2t and i=n-3t. Then U, V, W, m, and i satisfy the hypotheses of Lemma 9, which therefore shows that

$$|f_A(V)-f_A(W)| \leq t\delta(U)/(n-2t).$$

As in (a), the result follows because p and q were chosen arbitrarily.

(c) U and V satisfy the hypotheses of Lemma 4 and Lemma 7, and hence $f_M(V) \in \rho(U)$ and $f_A(V) \in \rho(U)$. This means that p's value just after round k is an element of $\rho(U)$. Since p is an arbitrary element of T, it follows that all values in U' are elements of $\rho(U)$, and hence $\rho(U')$ is contained in $\rho(U)$. □

Lemma 10 therefore shows that, at each round, the range of values of nonfaulty processes decreases by a factor of 1/2 in case $f = f_M$, and by a factor of $t/(n-2t)$ in case $f = f_A$. Thus the algorithm eventually converges; that is, the range of values held by nonfaulty processes eventually decreases to ε or better. If $3t+1 \leq n < 4t$, then f_M produces faster convergence, if $n > 4t$, then f_A produces faster convergence, and if $n=4t$, then both functions yield the same rate of convergence. Repeated application of part (c) of Lemma 10 shows that, at each round $k \geq 1$, the value held by nonfaulty processes immediately before round k are all in the range of initial values of nonfaulty processes. This shows that the algorithm satisfies the validity condition.

The above algorithm is still not complete, for as stated, it never terminates. Termination is achieved by the following strategy: 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 value 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 (1/2 if $f=f_M$, and $t/(n-2t)$ if $f=f_A$), and furthermore, it knows that the range of values it receives on the first round includes the initial values of all nonfaulty processes.

Each process can infer, after the first round, a round number at which the range of values of nonfaulty processes will be at most ε. In general, however, different processes will obtain different round numbers. Since all processes must halt at the same round in a synchronous approximation protocol, we must have some method of synchronizing the processes so that they agree to halt at the same round. This can be done using known solutions to the binary-valued Byzantine Generals problem as follows: After certain (a priori) selected rounds of the approximate agreement algorithm, all processes run a Byzantine Generals algorithm to decide whether the algorithm should continue. A process p will vote to halt only if the round number has reached the round number the p computed after the first round. Voting to halt implies that p knows that the range of values of nonfaulty processes is at most ε.

The proof of Theorem 1 is complete.

4. THE ASYNCHRONOUS PROBLEM

In this section, we reformulate the problem in an asynchronous model adapted from the one in [FLP 83]. Here, we assume that processes have states as before, but now the operation of the processes is described by a transition function which 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 messages. Nonfaulty processes always follow the protocol. 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 with the same outcome as for nonfaulty processes. However, they can change state arbitrarily (not necessarily according to the given protocol), and can send out any finite set of messages (not necessarily the ones specified by the protocol). A *T-computation* of an asynchronous system is one in which the processes in T always follow the protocol, all processes (faulty and nonfaulty) continue to take steps until they reach a halting state, and any process which fails to enter a halting state eventually receives all messages sent to it.

As asynchronous approximation protocol is said to be *t-correct* provided for every subset T of processes with $|T| \geq n-t$, in 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 to distinguish a faulty process from one which 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 protocol with n processes.

We now describe the asynchronous approximation protocol. As in the synchronous case, we first describe a nonterminating algorithm in which processes compute better and better approximations, then we discuss how termination is achieved. Assume that $n \geq 5t+1$. Let the function f be either f_M or f_A. At round k, each nonfaulty process p performs the following steps:

Asynchronous Approximation Algorithm:

1. Process p labels its current value with the current round number k, and then broadcasts this labeled value to all processes, including itself.
2. Process p waits to receive exactly $n-t$ round k 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 k values. Note that process p does *not* choose any default values, in contrast to the synchronous case.
3. Process p applies the function f to the multiset V to obtain its new value.

In analogy with Lemma 10, we have the following result, which states the convergence properties of the above algorithm.

Lemma 11: Assume that $n \geq 5t+1$. Let P be an asynchronous approximation protocol in which each process uses the Asynchronous Approximation Algorithm above. Suppose that T is a set of processes, with $|T| \geq n-t$. Let C be a T-computation of P and let k be a positive integer. Let U be the multiset of values held by processes in T immediately before round k in C, and let U' be the multiset of values held by processes in T immediately after round k in C. Then

(a) If $f = f_M$, then $\delta(U') \leq \delta(U)/2$.

(b) If $f = f_A$, then $\delta(U') \leq 2t\delta(U)/(n-3t)$.

(c) If either $f = f_A$ or $f = f_M$, then $\rho(U')$ is contained in $\rho(U)$.

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 k. 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 $|V \cap W| \geq n-3t > 2t$.

(a) U, V, and W satisfy the hypotheses of Lemma 6, which therefore shows that $|f_M(V)-f_M(W)| \leq \delta(U)/2$. Since p and q were chosen arbitrarily, the result follows.

(b) Let $m=n-3t$ and $i=n-5t$. Then U, V, W, m, and i satisfy the hypotheses of Lemma 9, which therefore shows that

$$|f_A(V)-f_A(W)| \leq 2t\delta(U)/(n-3t).$$

As in (a), the result follows because p and q were chosen arbitrarily.

(c) This part is identical to the proof of Lemma 10, part (c). □

Lemma 11 shows that, at each round, the range of values of nonfaulty processes decreases by a factor of 1/2 in case $f = f_M$, and by a factor of $2t/(n-3t)$ in case $f = f_A$. Thus the algorithm eventually converges. If $5t+1 \leq n < 7t$, then f_M produces faster convergence, if $n > 7t$, then f_A produces faster convergence, and if $n=7t$, then both functions yield the same rate of convergence. Repeated application of part (c) of Lemma 11 shows that, at each round $k \geq 1$, the value held by nonfaulty processes immediately before round k are all in the range of initial values of nonfaulty processes. This shows that the algorithm satisfies the validity condition.

The only remaining problem is termination. In the asynchronous case we cannot use simple Byzantine agreement on halting. Instead, we will use the following trick. We add 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 Agreement:

1. Process p labels its current value with the current 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 $\rho(\text{reduce}^2(V_p))$ (for definiteness, say $\text{mid}(\text{reduce}^2(V_p))$) 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 $k=2$). 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 values x_p for nonfaulty processes p;

(2) the value ε; and (3) the guaranteed rate of convergence (1/2 if $f = f_M$, and $2t/(n-3t)$ if $f = f_A$), 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.

Finally, we must do something about the fact that different processes will calculate different round numbers at which they would like to halt. To handle this, we modify the Asynchronous Agreement Algorithm in a simple way. Any process that reaches a round at which it wishes to halt, 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 original estimate). Although nonfaulty processes might obtain different estimates of the round at which the range of the values of nonfaulty processes is guaranteed to be sufficiently small, 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). Observe that a process can halt after it finds out that at least t+1 other processes had sent values with "halting" tags. The following lemma presents the above arguments in a precise way.

Lemma 12: Assume that $n \geq 5t+1$. Let P be an asynchronous approximation protocol in which each process uses the modified Asynchronous Approximation Algorithm above. Suppose that T is a set of processes, with $|T| \geq n-t$. Let C be a T-computation of P and let k be a positive integer such that some process in T has halted prior to the start of round k. Let U be the multiset of values held by processes in T immediately before round k in C, and let U' be the multiset of values held by processes in T immediately after round k in C. If either $f = f_A$ or $f = f_M$, then $\rho(U')$ is contained in $\rho(U)$.

Proof: Let p be an arbitrary processes in T. Let v and v' be the values held by p immediately before and after round k, respectively. It suffices, since p is arbitrary, to show that $v' \in \rho(U)$. If p has terminated prior to the start of round k, then $v' = v \in \rho(U)$. If p has not halted prior to the start of round k, then let V be the multiset of values received by p in round k. Then V and U satisfy the hypotheses of Lemma 4 and Lemma 7, and since either $v' = f_M(V)$ or $v' = f_A$, it follows that $v' \in \rho(U)$. □

Thus the range of values of nonfaulty processes never increases once some nonfaulty process has halted. The proof of Theorem 2 is complete.

5. On Resiliency of the Approximation Algorithms

Because the number of rounds required for convergence of the above approximation algorithms depends on the values of all the processes at the first round, faulty processes can force a large number of rounds. Processes may fail and recover several times throughout the algorithm; consequently, it may be unduly cautious to require that t be the upper bound on the total number of faulty processes during the entire algorithm. To sustain more than t total faults, it is desirable that a recovered process be able to reintegrate into the algorithm as a nonfaulty process. In particular, it should be able to resume correctly within a few rounds.

In the sequel we do not give a very precise and formal model for resiliency and reintegration. Rather, we discuss the issue of reintegration and indicate how to obtain better resiliency. Observe that when a recovered process is reintegrated and is nonfaulty, another process may fail without increasing the bound on the number of processes that are concurrently faulty.

We say that a process p at configuration C_i is in a *faulty mode* if either the transition from C_i to C_{i+1} is not according to the transition function, or the messages it generates after C_i are not according to the message generation function. The definition of a T-computation can be refined to distinguish at every round between faulty and nonfaulty processes. After a process switches from faulty to nonfaulty mode, it is called *recovered*.

In general the transition of a process in a faulty mode is completely unpredictable. Allowing a faulty process to enter arbitrary states may imply limitations on the reintegration of faulty processes. Since we require that a process stay in a final state once such a state is reached and since faulty transitions into final states are not excluded, no algorithm can guarantee that processes which are only temporarily faulty will always be reintegrated later. A faulty process that incorrectly enters a halting state must be considered faulty for the rest of the algorithm.

Since a previously faulty process might not be able to reconstruct its state, the reintegration of a recovered process can present a problem. Ideally, a previously faulty process returns to some new state from which it can continue to function as a nonfaulty process. In our algorithms a process is considered reintegrated if it returns to such a state with a value in the range of values of the nonfaulty processes. If this happens we say that the recovered process resumes in a nonfaulty state. In our algorithms a recovered process needs must be able to participate correctly in an ongoing Byzantine agreement algorithm. In particular, it must decide on a round from which it will support halting.

5.1. The synchronous case

First consider approximation algorithms for the synchronous case. If such an algorithm uses only the Synchronous Approximation Algorithm repeatedly, a recovered process can resume in a nonfaulty state only one round after recovery. Since, during that round its value may influence the convergence of the approximation functions. It should be counted as one of the faulty processes with respect to the upper bound t.

Let U be a multiset with $|U|>2t$, and define $f_\mu(U) = \min(s^t(U))$. Assume that at every round each process is required to send not only its value but also its estimate of the number of rounds that still remain. One can prove that by using f_μ a recovered process can obtain a correct estimate for the number of rounds remaining. Moreover, at every round (starting at the second round) each process can use f_μ to obtain a new estimate for the number of rounds that still remain. If U is the multiset of estimates of the number of remaining rounds, then, $f_\mu(U)-1$ is the new estimate that a process can use. It can be shown that by using this method we do not increase the total number of rounds beyond that given in Section 3.

None of the known Byzantine agreement algorithms can permit a failed process to become nonfaulty only one round after recovery. Since our synchronous approximation algorithm requires the use of a Byzantine agreement algorithm for its termination, it inherits this deficiency. The Byzantine agreement algorithm introduced in [R 83] can be used to allow a process to resume in a nonfaulty state within three rounds of recovery; however, this algorithm guarantees only that all nonfaulty processes halt within two consecutive rounds. Use of this algorithm to achieve termination therefore results in a synchronous approximation algorithm that is not t-correct according to our original definition, but which satisfies the somewhat weaker termination condition stated below.

An approximation algorithm is called *semi-synchronous* if it satisfies all the conditions of a synchronous algorithm which is t-correct except for the condition that all nonfaulty processes halt at the same round. The approximation algorithm obtained by sending the estimates for the remaining round, using the above f_μ, and using the algorithm of [R 83] to achieve termination is a semi-synchronous t-correct approximation algorithm. In that algorithm a recovered process can resume in a nonfaulty state and become nonfaulty within three rounds. Thus, for each round t is an upper bound for the total number of processes that are faulty at that round together with processes that have recovered within the previous two rounds.

5.2. The asynchronous case

We are using a very general model for the asynchronous case. The asynchronous nature of the communication medium prevents us from using incoming messages to reintegrate recovered processes within a constant number of rounds. The nondeterministic behavior of the communication medium can increase arbitrarily the time it takes a recovered process to reintegrate. To obtain bounded results a more restricted model of the communication medium must be used. We are currently analyzing the tradeoff between the restriction on the model and the resiliency of the approximation algorithm. These results will appear in a future version of this paper.

6. SUMMARY AND OPEN QUESTIONS

A problem of approximate agreement on real numbers by processes in a distributed system has been defined. In addition, two simple approximation functions have been integrated into two simple-to-implement algorithms for achieving approximate agreement – one for a synchronous distributed system, and the other for an asynchronous distributed system. The algorithm for an asynchronous system encapsulates new ideas for the design of algorithms that are not vulnerable to delays and never wait indefinitely.

This paper presents two examples of approximation functions, which yield differing convergence factors for different values of n and t. In the synchronous case, the function f_M is preferable over the range $3t+1 \le n < 4t$, where it produces a convergence factor of 1/2, and the function f_A is preferable over the range $n \ge 4t+1$, where it produces a convergence factor of $t/(n-2t)$. For $n=4t$, both functions produce a convergence factor of 1/2. The situation is similar in the asynchronous case: If $5t+1 \le n < 7t$, then f_M produces faster convergence; if $n > 7t$, then f_A produces faster convergence; if $n=7t$, then both functions produce a convergence factor of 1/2.

An interesting question is whether any other approximation functions can produce faster convergence than the above functions. We have been able to prove that, among a large class of functions, the single-round convergence factor of 1/2 produced by the function f_M is optimal over the range $3t+1 \le n \le 4t$ for the synchronous case and $5t+1 \le n \le 7t$ for the asynchronous case. Also, the single-round convergence factor of $t/(n-2t)$ produced by the function f_A is optimal in the synchronous case when $n=kt$, where $k \ge 4$. Similarly, the single-round convergence factor of $2t/(n-3t)$ produced by f_A is optimal in the asynchronous case when $n=(2k+1)t$, where $k \ge 3$.

It seems that the ideas of this paper can be applied to the problem of clock synchronization [LM 82].

We do not yet know if the requirement that n be at least 5t+1 is necessary for the asynchronous case. For the synchronous case we can show that 3t+1 is necessary if there is no authentication scheme. The proof is an adaptation of the lower bound proof in [LSP 82].

We conjecture that there is no t-correct synchronous approximation algorithm with no assumptions about recovered processes in which recovered processes become nonfaulty within one round. We also conjecture that to obtain synchronous halting one will need additional halting condition, such as: there exists some k consecutive rounds with at most k-1 faulty processes in all of them.

Other open questions involve upper and lower bounds on the complexity of the approximate agreement problem, where the complexity measures are time, number of messages, and total number of bits transmitted.

REFERENCES

[DDS] D. Dolev, C. Dwork and L. Stockmeyer, "On the Minimal Synchronism Needed for Distributed Consensus," 24th Annual Symposium on Foundations of Computer Science, Nov. 1983.

[DS] 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.

[FLP] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with one faulty process," proceedings, the 2nd ACM Symposium on Principles of Database Systems, 1983.

[LM] L. Lamport and P. M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults," Technical Report, Computer Science Laboratory, SRI International, March 1982.

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

[R] R. Reischuk, "A New Solution for the Byzantine Generals Problem," proceedings, Symposium on Foundations of Computer Science, Sweden, Aug. 1983.