Reaching Agreement in the Presence of Faults

M. PEASE, R. SHOSTAK, AND L. LAMPORT

SRI International, Menlo Park, California

ABSTRACT. The problem addressed here concerns a set of isolated processors, some unknown subset of which may be faulty, that communicate only by means of two-party messages. Each nonfaulty processor has a private value of information that must be communicated to each other nonfaulty processor. Nonfaulty processors always communicate honestly, whereas faulty processors may lie The problem is to devise an algorithm in which processors communicate their own values and relay values received from others that allows each nonfaulty processor to infer a value for each other processor The value inferred for a nonfaulty processor must be that processor's private value, and the value inferred for a faulty one must be consistent with the corresponding value inferred by each other nonfaulty processor

It is shown that the problem is solvable for, and only for, $n \geq 3m + 1$, where m is the number of faulty processors and n is the total number. It is also shown that if faulty processors can refuse to pass on information but cannot falsely relay information, the problem is solvable for arbitrary $n \geq m \geq 0$. This weaker assumption can be approximated in practice using cryptographic methods

KEY WORDS AND PHRASES. agreement, authentication, consistency, distributed executive, fault avoidance, fault tolerance, synchronization, voting

CR CATEGORIES: 3.81, 4.39, 5.29, 5.39, 6.22

1. *Introduction*

Fault-tolerant systems often require a means by which independent processors or processes can arrive at an exact mutual agreement of some kind. It may be necessary, for example, for the processors of a redundant system to synchronize their internal clocks periodically. Or they may have to settle upon a value of a time-varying input sensor that gives each of them a slightly different reading. In the absence of faults reaching a satisfactory mutual agreement is usually an easy matter. In most cases it suffices simply to exchange values (times, in the case of clock synchronization) and compute some kind of average. In the presence of faulty processors, however, simple exchanges cannot be relied upon; a bad processor might report one value to a given processor and another value to some other processors, causing each to calculate a different "average."

One might imagine that the effects of faulty processors could be dealt with through the use of voting schemes involving more than one round of information exchange; such schemes might force faulty processors to reveal themselves as faulty or at least to behave consistently enough with respect to the nonfaulty processors to allow the latter to reach an exact agreement. As we will show, it is not always possible to devise schemes of this kind, even if it is known that the faulty processors are in a minority. Algorithms that allow exact agreement to be reached by the nonfaulty processors do exist, however, if they sufficiently outnumber the faulty ones.

Our results are formulated using the notion of *interactive consistency*, which we define

This work was supported in part by NASA-Langley Research Center under Contract NASI-13792 and by the Ballistic Missile Defense Advanced Technology Center under Contract DASG60-78-C-0046

Authors' address Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025

as follows: Consider a set of n isolated processors, of which it is known that no more than m are faulty. It is not known, however, which processors are faulty. Suppose that the processors can communicate only by means of two-party messages. The communication medium is presumed to be fail-safe and of negligible delay. The sender of a message, moreover, is always identifiable by the receiver. Suppose also that each processor p has some private value of information V_p (such as its clock value or its reading of some sensor). The question is whether for given m, $n \geq 0$, it is possible to devise an algorithm based on an exchange of messages that will allow each nonfaulty processor p to compute a vector of values with an element for each of the n processors, such that

(1) the nonfaulty processors compute exactly the same vector;
(2) the element of this vector corresponding to a given nonfaulty processor is the private value of that processor.

Note that the algorithm need not reveal which processors are faulty, and that the elements of the computed vector corresponding to faulty processors may be arbitrary; it matters only that the nonfaulty processors compute exactly the same value for any given faulty processor.

We say that such an algorithm achieves interactive consistency, since it allows the nonfaulty processors to come to a consistent view of the values held by all the processors, including the faulty ones. The computed vector is called an *interactive consistency* (*i.c.*) *vector*. Once interactive consistency has been achieved, each nonfaulty processor can apply an averaging or filtering function to the i.c. vector, according to the needs of the application. Since each nonfaulty processor applies this function to the same vector of values, an exact agreement is necessarily reached.

We show in the following sections that algorithms can be devised to guarantee interactive consistency for and only for n, m such that $n \geq 3m + 1$. It will follow, in particular, that a minimum of four processors is required in the single-fault case. We also show, however, that interactive consistency can be assured for arbitrary $n \geq m \geq 0$ if it is assumed that faulty processors can refuse to pass on information obtained from other processors but cannot falsely report this information. This assumption can be approximated in practice using *authenticators*, which we discuss in Section 5.

We begin in Section 2 with a description of the single-fault case. Section 3 is concerned with the generalization to $n \geq 3m + 1$ and Section 4 with an impossibility argument for $n \leq 3m$. Section 5 gives an algorithm for arbitrary $n \geq m \geq 0$ that works under the restricted assumption stated above. Conclusions and issues for future study are given in Section 6.

Problems similar to the one considered here have been studied by Davies and Wakerly [1].

2. *The Single-Fault Case*

In order to give the reader a feeling for the problem, we begin with a procedure for obtaining interactive consistency in the simple case of $m = 1$, $n = 4$. The procedure consists of an exchange of messages, followed by the computation of the interactive consistency vector on the basis of the results of the exchange.

Two rounds of information exchange are required. In the first round the processors exchange their private values. In the second round they exchange the results obtained in the first round. The faulty processor (if there is one) may "lie," of course, or refuse to send messages. If a nonfaulty processor p fails to receive a message it expects from some other processor, p simply chooses a value at random and acts as if that value had been sent.

The exchange having been completed, each nonfaulty processor p records its private value V_p for the element of the interactive consistency corresponding to p itself. The element corresponding to every other processor q is obtained by examining the three received reports of q's value (one of these was obtained directly from q in the first round, and the others from the remaining two processors in the second round). If at least two of

the three reports agree, the majority value is used. Otherwise, a default value such as "NIL" is used.

To see that this procedure assures interactive consistency, first note that if q is nonfaulty, p will receive V_q both from q and from the other nonfaulty processor(s). Thus p will record V_q for q as desired. Now suppose q is faulty. We must show only that p and the other two nonfaulty processors record the same value for q. If every nonfaulty processor records NIL, we are done. Otherwise, some nonfaulty processor, say p, records a non-NIL value v, having received v from at least two other processors. Now if p received v from both of the other nonfaulty processors, each other nonfaulty processor must receive v from every processor other than p (and possibly from p as well); every nonfaulty processor will thus record v. Otherwise, p must have received v from all processors other than some other nonfaulty processor p'. In this case p' received v from all processors other than q (so p' records v), and all other nonfaulty processors received v from all processors other than p'. All nonfaulty processors therefore record v as required.

3. *A Procedure for* $n \geq 3m + 1$

Recall that the procedure given in the last section requires two rounds of information exchange, the first consisting of communications of the form "my private value is" and the second consisting of communications of the form "processor x told me his private value is ...." In the general case of m faults, $m + 1$ rounds are required. In order to describe the algorithm, it will be convenient to characterize this exchange of messages in a more formal way.

Let P be the set of processors and V a set of values. For $k \geq 1$, we define a *k-level scenario* as a mapping from the set of nonempty strings (possibly having repetitions) over P of length $\leq k + 1$, to V. For a given k-level scenario σ and string $w = p_1 p_2 \cdots p_r$, $2 \leq r \leq k + 1$, $\sigma(w)$ is interpreted as the value p_2 tells p_1 that p_3 told p_2 that p_4 told p_3 ... that p_r told p_{r-1} is p_r's private value. For a single-element string p, $\sigma(p)$ simply designates p's private value V_p. A k-level scenario thus summarizes the outcome of a k-round exchange of information. (Note that if a faulty processor lies about who gave it information, this is equivalent to lying about a value it was given.) Note also that for a given subset of nonfaulty processors, only certain mappings are possible scenarios; in particular, since nonfaulty processors are always truthful in relaying information, a scenario must satisfy

$$\sigma(pqw) = \sigma(qw)$$

for each nonfaulty processor q, arbitrary processor p, and string w.

The messages a processor p receives in a scenario σ are given by the restriction σ_p of σ to strings beginning with p. The procedure we present now for arbitrary $m \geq 0$, $n \geq 3m + 1$, is described in terms of p's computation, for a given σ_p, of the element of the interactive-consistency vector corresponding to each processor q. The computation is as follows:

(1) If for some subset Q of P of size $>(n + m)/2$ and some value v, $\sigma_p(pwq) = v$ for each string w over Q of length $\leq m$, p records v.

(2) Otherwise, the algorithm for $m - 1$, $n - 1$ is recursively applied with P replaced by $P - \{q\}$, and σ_p by the mapping $\hat{\sigma}_p$ defined by

$$\hat{\sigma}_p(pw) = \sigma_p(pwq)$$

for each string w of length $\leq m$ over $P - \{q\}$. If at least $\lfloor (n + m)/2 \rfloor$ of the $n - 1$ elements in the vector obtained in the recursive call agree, p records the common value, otherwise p records NIL.

Note that $\hat{\sigma}_p$ corresponds to the m-level subscenario of σ in which q is excluded and in which each processor's private value is the value it obtains directly from q in σ. Note also that the algorithm essentially reduces to the one given in the last section in the case $m = 1$, $n = 4$.

The proof that the algorithm given above does indeed assure interactive consistency proceeds by induction on m:

Basis $m = 0$. In this case no processor is faulty, and the algorithm always terminates in step (1) with p recording V_q for q.

Induction Step $m > 0$. First note that if q is nonfaulty, $\sigma_p(pwq) = V_q$ for each string w (including the empty string) of length $\leq m$ over the set of nonfaulty processors. This set has $n - m$ members (which, since $n > 3m$, is $>(n + m)/2$) and so satisfies the requirements for Q in step (1) of the algorithm. Any other set satisfying these requirements, moreover, must contain a nonfaulty processor (since it must be of size $>(n + m)/2$, and $n \geq 3m + 1$) and must therefore also yield V_q as the common value. The algorithm thus terminates at step (1), and p records V_q and q as required.

Now suppose that q is faulty. We must show that the value p records for q agrees with the value each other nonfaulty processor p' records for q.

First consider the case in which both p and p' exit the procedure at step (1), each having found an appropriate set Q. Since each such set has more than $(n + m)/2$ members, and since P has only n members in all, the two sets must have more than $2((n + m)/2) - n = m$ common members. Since at least one of these must be nonfaulty, the two sets must give rise to the same value v, as required.

Next suppose that p' exits at step (1), having found an appropriate set Q and common value v, and that p executes step (2). We claim that in the vector of $n - 1$ elements that p computes in the recursive call, the elements corresponding to members of $\hat{Q} = Q - \{q\}$ are equal to v. Since $\hat{Q}$ has at least $\lfloor (n + m)/2 \rfloor$ members, it will then follow that p records v in accordance with step (2). To see that the elements corresponding to members of $\hat{Q}$ are indeed equal to v, recall that the mapping $\hat{\sigma}_p$ that p uses to compute the vector in the recursive call is the restriction, to strings beginning with p, of the m-level scenario $\hat{\sigma}_p$ defined by

$$\hat{\sigma}_p(w) = \sigma(wq)$$

for each string w of length $\leq m$ over $P - \{q\}$. By the induction hypothesis, this vector is identical to the one p' would have computed using the restriction $\hat{\sigma}_{p'}$ of $\hat{\sigma}$ had p' made the recursive call. Moreover, the value p' would have computed for the element of this vector corresponding to a given q' in $\hat{Q}$ must be v, since $\hat{Q}$ and v satisfy step (1) of the algorithm. (Note that $\hat{Q}$ is of size $\geq \lfloor (n + m)/2 \rfloor > [(n - 1) + (m - 1)]/2$, and that $\sigma_{p'}(p'wq') = \sigma_{p'}(p'wq'q) = v$ for each string w of length $\leq m - 1$ over $\hat{Q}$.) The case in which p exits at step (1) and p' exits at step (2) is handled similarly.

In the one remaining case, both p and p' exit at step (2). In this case both recurse and must, by the induction hypothesis, compute exactly the same vector, and hence the same value for q. Q.E.D.

4. *Proof of Impossibility for* $n < 3m + 1$

The procedure of the last section guarantees interactive consistency only if $n \geq 3m + 1$. In this section it is shown that the $3m + 1$ bound is tight. We will prove not only that it is impossible to assure interactive consistency for $n < 3m + 1$ with $m + 1$ rounds of information exchange, but also that it is impossible, even allowing an infinite number of rounds of exchange (i.e., using scenarios mapping from *all* nonempty strings over P to V).

Just to gain some intuitive feeling as to why $3m$ processors are not sufficient, consider the case of three processors A, B, C, of which one, say C, is faulty. By prevaricating in just the right way, C can thwart A's and B's efforts to obtain consistency. In particular, C's messages to A can be such as to suggest to A that C's private value is, say, 1, and that B is faulty. Similarly, C's messages to B can be such as to suggest to B that C's private value is 2, and that A is faulty. If C plays its cards just right, A will not be able to tell whether B or C is faulty, and B will not be able to tell whether A or C is at fault. A will thus have no

choice but to record 1 for C's value, while B must record 2, defeating interactive consistency.

In order to give a precise statement of the impossibility result and its proof, a few formal definitions are needed.

First, define a *scenario* as a mapping from the set P^+ of all nonempty strings over P, to V. For a given $p \in P$ define a *p-scenario* as a mapping from the subset of P^+, consisting of strings beginning with p, to V.

Next, for a given choice $N \subseteq P$ of nonfaulty processors and a given scenario σ, say that σ is *consistent with* N if for each $q \in N$, $p \in P$, and $w \in P^*$ (set of all strings over P), $\sigma(pqw) = \sigma(qw)$. (In other words, σ is consistent with N if each processor in N always reports what it knows or hears truthfully.)

Now define the notion of interactive consistency as follows. For each $p \in P$, let F_p be a mapping that takes a p-scenario σ_p and a processor q as arguments and returns a value in V. (Intuitively, F_p gives the value that p computes for the element of the interactive consistency vector corresponding to q on the basis of σ_p.) We say that $\{F_p | p \in P\}$ assures *interactive consistency for m faults* if for each choice of $N \subseteq P$, $|N| \geq n - m$, and each scenario σ consistent with N,

(i) for all $p, q \in N$, $F_p(\sigma_p, q) = \sigma(q)$,
(ii) for all $p, q \in N$, $r \in P$, $F_p(\sigma_p, r) = F_q(\sigma_q, r)$,

where σ_p and σ_q denote the restrictions of σ to strings beginning with p and q, respectively.

Intuitively, clause (i) requires that each nonfaulty processor p correctly compute the private value of each nonfaulty processor q, and clause (ii) requires that each two nonfaulty processors compute exactly the same vector.

THEOREM. *If $|V| \geq 2$ and $n \geq 3m$, there exists no $\{F_p | p \in P\}$ that assures interactive consistency for m faults.*

PROOF. Suppose, to the contrary, that $\{F_p | p \in P\}$ assures interactive consistency for m faults. Since $n \leq 3m$, P can be partitioned into three nonempty sets A, B, and C, each of which has no more than m members. Let v and v' be two distinct values in V. Our general plan is to construct three scenarios α, β, and σ such that α is consistent with $N = A \cup C$, β with $N = B \cup C$, and σ with $N = A \cup B$. The members of C will all be given private value v in α and v' in β. Moreover, α, β, and σ will be constructed in such a way that no processor $a \in A$ can distinguish α from σ (i.e., $\alpha_a = \sigma_a$), and no processor $b \in B$ can distinguish β from σ (i.e., $\beta_b = \sigma_b$). It will then follow that for the scenario σ processors in A and B will compute different values for the members of C.

We define the scenarios α, β, and σ recursively as follows:

(i) For each $w \in P^+$ not ending in a member of C, let

$$\alpha(w) = \beta(w) = \sigma(w) = v.$$

(ii) For each $a \in A$, $b \in B$, $c \in C$ let

$$\begin{aligned} &\alpha(c) = \alpha(ac) = \alpha(bc) = \alpha(cc) = v, \\ &\beta(c) = \beta(ac) = \beta(bc) = \beta(cc) = v', \\ &\sigma(c) = \sigma(ac) = \sigma(cc) = v, \qquad \sigma(bc) = v'. \end{aligned}$$

(iii) For each $a \in A$, $b \in B$, $c \in C$, $p \in P$, $w \in P^*c$ (i.e., w is any string over P ending in c), let

$$\begin{aligned} &\alpha(paw) = \alpha(aw), \quad &&\beta(paw) = \alpha(aw), \\ &\alpha(pbw) = \beta(bw), \quad &&\beta(pbw) = \beta(bw), \\ &\alpha(pcw) = \alpha(cw), \quad &&\beta(pcw) = \beta(cw), \end{aligned}$$

$$\begin{aligned} \sigma(paw) &= \sigma(aw), \\ \sigma(pbw) &= \sigma(bw), \\ \sigma(acw) &= \alpha(cw), \\ \sigma(bcw) &= \beta(cw). \end{aligned}$$

It is easy to verify by inspection that α, β, and σ are in fact consistent with $N = A \cup C$, $B \cup C$, $A \cup B$, respectively. Moreover, one can show by a simple induction proof on the length of w that

$$\alpha(aw) = \sigma(aw), \qquad \beta(bw) = \sigma(bw)$$

for all $a \in A$, $b \in B$, and $w \in P^*$.

It then follows from the definition of interactive consistency that for any $a \in A$, $b \in B$, $c \in C$,

$$v = \alpha(c) = F_a(\alpha_a, c) = F_a(\sigma_a, c) = F_b(\sigma_b, c) = F_b(\beta_b, c) = v',$$

giving a contradiction. Q.E.D.

5. *An Algorithm Using Authenticators*

The negative result of the last section depends strongly on the assumption that a faulty processor may refuse to pass on values it has received from other processors or may pass on fabricated values. This section addresses the situation in which the latter possibility is precluded. We will assume, in other words, that a faulty processor may "lie" about its own value and may refuse to relay values it has received, but may not relay altered values without betraying itself as faulty.

In practice, this assumption can be satisfied to an arbitrarily high degree of probability using *authenticators*. An authenticator is a redundant augment to a data item that can be created, ideally, only by the originator of the data. A processor p constructs an authenticator for a data item d by calculating $A_p[d]$, where A_p is some mapping known only to p. It must be highly improbable that a processor q other than p can generate the authenticator $A_p[d]$ for a given d. At the same time, it must be easy for q to check, for a given p, v, and d, that $v = A_p[d]$. The problem of devising mappings with these properties is a cryptographic one. Methods for their constructions are discussed in [2] and [3]. For many applications in which faults are due to random errors rather than to malicious intelligence, any mappings that "suitably randomize" the data suffice.

A scenario σ is carried out in the following way. As before, let $v = \sigma(p)$ designate p's private value. p communicates this value to r by sending r the message consisting of the triple $\langle p, a, v \rangle$, where $a = A_p[v]$. When r receives the message, it checks that $a = A_p[v]$. If so, r takes v as the value of $\sigma(rp)$. Otherwise r lets $\sigma(rp) = \text{NIL}$. More generally, if r receives exactly one message of the form $(p_1, a_1(p_2, a_2 \ldots (p_k, a_k, v) \ldots))$, where $a_k = A_k[v]$ and for $1 \leq i \leq k - 1$, $a_i = A_i[(p_{i+1}, a_{i+1} \ldots (p_k, a_k, v)]$, then $\sigma(rp_1 \cdots p_k) = v$. Otherwise, $\sigma(rp_1 \cdots p_k) = \text{NIL}$.

A scenario σ constructed in this way is consistent with a given choice N of nonfaulty processors, if for all processors $p \in N$, $q \in P$ and strings w, w' over P.

(i) $\sigma(qpw) = \sigma(pw)$,
(ii) $\sigma(w'pw)$ is either $\sigma(pw)$ or NIL.

Condition (i) ensures that nonfaulty processors are always truthful. Condition (ii) guarantees that a processor cannot relay an altered value of information received from a nonfaulty processor.

We now present a procedure, using $m + 1$-level authenticated scenarios, that guarantees interactive consistency for any $n \geq m$. As before, the procedure is described in terms of the value a nonfaulty processor p records for a given processor q on the basis of σ_p:

> Let S_{pq} be the set of all non-NIL values $\sigma_p(pwq)$, where w ranges over strings of distinct elements with length $\leq m$ over $P - \{p, q\}$. If S_{pq} has exactly one element v, p records v for q; otherwise, p records NIL.

To see that interactive consistency is assured, consider first the case in which q is nonfaulty. In this case $\sigma_p(pwq)$ is either $\sigma(q)$ or NIL for each appropriate w by condition (ii). Since, in particular, $\sigma_p(pq) = \sigma(q)$ by (i), $S_{pq} = \{\sigma(q)\}$. p thus records $\sigma(q)$ for q as required.

If q is faulty, it suffices to show only that for each two nonfaulty processors p and p', $S_{pq} = S_{p'q}$. So suppose $v \in S_{pq}$, i.e., $v = \sigma_p(pwq)$ for some string w having no repetitions, with length $\leq m$ over $P - \{p, q\}$. If p' occurs in w (say $w = w_1p'w_2$), then $\sigma(pwq) = \sigma(p'w_2q)$ by (ii); hence $v = \sigma(pwq) \in S_{p'q}$. If p' does not occur in w and w is of length $<m$, then pw is of length $\leq m$; so $v = \sigma(pwq) = \sigma(p'pwq) \in S_{p'q}$. Finally, if p' does not occur in w and w is of length m, w must be of the form w_1rw_2 where r is nonfaulty, giving that $v = \sigma(pwq) = \sigma(rw_2q)$ (by (ii)) $= \sigma(p'rw_2q)$ (by (i)) $\in S_{p'q}$. In each case $v \in S_{p'q}$. A symmetrical argument shows that if $v \in S_{p'q}$, $v \in S_{pq}$. Hence $S_{p'q} = S_{pq}$ as required. Q.E.D.

6. *Conclusions*

The problem of obtaining interactive consistency appears to be quite fundamental to the design of fault-tolerant systems in which executive control is distributed. In the SIFT [4] fault-tolerant computer under development at SRI, the need for an interactive consistency algorithm arises in at least three aspects of the design—synchronization of clocks, stabilization of input from sensors, and agreement on results of diagnostic tests. In the preliminary stages of the design of this system, it was naively assumed that simple majority voting schemes could be devised to treat these situations. The gradual realization that simple majorities are insufficient led to the results reported here.

These results by no means answer all the questions one might pose about interactive consistency. The algorithms presented here are intended to demonstrate existence. The construction of efficient algorithms and algorithms that work under the assumption of restricted communications is a topic for future research. Other questions that will be considered include those of reaching approximate agreement and reaching agreement under various probabilistic assumptions.

ACKNOWLEDGMENTS. The authors gratefully acknowledge the substantial contribution of ideas to this paper by K. N. Levitt, P. M. Melliar-Smith, and J. H. Wensley, and the reviewers. We are especially grateful to E. Migneault of NASA-Langley for his perceptive insights into the importance and difficulty of the problem.

REFERENCES

1. DAVIES, D., AND WAKERLY, J. Synchronization and matching in redundant systems *IEEE Trans on Comptrs. C-27*, 6 (June 1978), 531–539.
2. DIFFIE, W., AND HELLMAN, M. New directions in cryptography. *IEEE Trans Inform. Theory IT-22*, 6 (Nov 1976), 644–654
3. RIVEST, R.L., SHAMIR, A., AND ADLEMAN, L A A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 2 (Feb 1978), 120–126.
4. WENSLEY, J H., ET AL. SIFT: design and analysis of a fault-tolerant computer for aircraft control *Proc. IEEE 66*, 10 (Oct. 1978), 1240–1255.

RECEIVED NOVEMBER 1978; REVISED APRIL 1979; ACCEPTED MAY 1979