Randomized Asynchronous Byzantine Agreements

Sam Toueg

TR 83-587
December 1983
(Revised February 1984)

Department of Computer Science
Cornell University
Ithaca, New York 14853

Randomized Asynchronous Byzantine Agreements

Sam Toueg

Cornell University
Ithaca, New York 14853

ABSTRACT

A randomized protocol for reaching Byzantine Agreement in asynchronous systems with n processes was recently proposed in [Rabi83]. This protocol tolerates up to $\lfloor (n-1)/10 \rfloor$ faulty processes, and agreement is reached within an expected number of phases that is a small constant independent of n and the number of faulty processes t. In this paper, using the same computation model as in [Rabi83], it is shown that no Byzantine Agreement protocol can overcome more than $\lfloor (n-1)/3 \rfloor$ faulty processes in an asynchronous system, and we describe a protocol that achieves this upper bound. Agreement is also reached within an expected number of phases that is a small constant independent of n and t, but the communication complexity is higher than in [Rabi83].

1. Introduction

We consider a set of n communicating processes $\{ G_1, \ldots, G_n \}$ with initial values $\{ M_1, \ldots, M_n \}$. The processes execute an *agreement* protocol to agree on an identical message value. *Proper* processes faithfully follow the protocol, while *faulty* processes deviate from it, maybe in a malicious effort to prevent agreement. We say that a system is *proper with initial value M* if all the proper processes have the initial value M. A *Byzantine Agreement* protocol is an agreement protocol with the additional constraint that if the system is proper with initial value M then the proper processes must agree on M at the end of the protocol.

The problem of reaching agreement in the presence of faults is widely recognized as a fundamental problem in fault-tolerant computing, and it has been extensively studied in the last five years [Peas80, Dole81, Dole82, Lamp82, Lync82, Dole83]. Most protocols proposed so far apply only to synchronous systems. Moreover, Fischer et al. [Fisc83] showed that asynchronous systems have no deterministic agreement protocols that always terminate.

Following this result, probabilistic or randomized Byzantine Agreement protocols that terminate with probability 1 were developed [Brac83, Brac84, BenO83], but if the number of faulty processes is greater than $O(\sqrt{n})$ these protocols require a large (i.e., $\Omega(2^n)$) expected number of phases† to terminate.

Another randomized Byzantine Agreement protocol was proposed by Rabin [Rabi83]. It differs from [Brac83, Brac84, BenO83] by assuming messages are authenticated by digital signatures, and processes share a secret sequence of random bits that is supplied *in advance* by a non-faulty dealer. This protocol tolerates up to $\lfloor (n-1)/10 \rfloor$ faulty processes, and agreement is reached within an expected number of phases that is a small constant independent of n and the number of faulty processes t.

In this paper, using the same computation model as in [Rabi83], it is shown that no Byzantine Agreement protocol can overcome more than $\lfloor (n-1)/3 \rfloor$ faulty processes in an asynchronous system, and we describe a protocol that achieves this upper bound. Agreement is also reached within an expected number of phases that is a small constant independent of n and t, but the communication complexity is higher than in [Rabi83].

The remainder of this paper is organized as follows. In Section 2, we briefly summarize the main assumptions of our model. In Section 3, we describe a broadcast protocol that is later used as the underlying broadcast primitive in our Byzantine Agreement protocols. A randomized Byzantine Agreement protocol for binary valued messages is presented in Section 4. This protocol terminates within a fixed number of phases, with a small probability of error. In Section 5, the protocol is modified into an errorless protocol that terminates within a small expected number of phases. We show how to extend the above protocols to multivalued messages in Section 6. Finally, in Section 7, we show the optimality of the proposed protocols with respect to the number of faulty processes that they can overcome.

2. The model assumptions

We use a model identical to the one proposed by Rabin in [Rabi83], and, in particular, we have the same assumptions.

As mentioned earlier, the main assumption is the existence of a non-faulty dealer that generates a sequence of random bits $s_1, s_2, \ldots, s_k, \ldots,$ and uses Shamir's algorithm [Sham79] to make each bit a shared secret among the n processors. In fact, for each bit s_k,

† Informally, a phase consists of a communication round where each process broadcasts a message, and then waits to receive messages sent by the other processes in the same round.

the dealer computes n pieces of the shared secret s_k such that the knowledge of any $t + 1$ such pieces is necessary and sufficient for computing s_k. The dealer signs all the n pieces to prevent forgery, and it distributes them among the n processes. As mentioned earlier, this is done for all the bits s_k, prior to the execution of the agreement protocol.

Based on the above assumption, we postulate the *compute_secret*(s_k) procedure. When a process executes *compute_secret*(s_k)

1. it broadcasts its piece of the shared secret bit s_k to all the processes, and
2. it waits to receive $t+1$ such pieces, and then computes s_k.

The other two assumptions made in [Rabi83] and in this paper are

(i) failure-free communication exists between any two processor, and

(ii) digital signatures uniquely identify the originator of each message.

3. The broadcast protocol

In this section, we describe a broadcast protocol that effectively limits the power of faulty processes during a broadcast. The protocol can be applied to asynchronous systems with $n \geq 3t + 1$.

When a process G broadcasts a message m the following protocol is followed:

1. G sends a [*initial*, G, m] message to all the processes.
2. When a process receives the first [*initial*, G, m] message from G, it sends a [*echo*, G, m] message to all the processes. Any subsequent [*initial*, G, m'] message is ignored.
3. If a process receives an [*echo*, G, m] message from more than $(n+t)/2$ distinct processes, then it *accepts* the message m from G.

Lemma 1 [Brac83]: If a process G broadcasts the message m then

1. The messages accepted by proper processes are identical, and
2. if G is proper, then all the proper processes accept m from G.

(Note that if G is faulty, then the following can occur: some proper processes accept a message from G, and some other proper processes do not. The first part of the lemma only guarantees that all the accepted messages are identical.)

Proof:

1. The proof of the first part is by contradiction. Suppose that two proper processes G_1 and G_2 accept two different messages m_1 and m_2. Then, more than $(n+t)/2$ processes sent $[echo, G, m_1]$ messages (to G_1), and more than $(n+t)/2$ processes sent $[echo, G, m_2]$ messages (to G_2). Since there are only n processes in the system, then more than t processes sent both $[echo, G, m_1]$ and $[echo, G, m_2]$ messages. Such processes deviated from the broadcast protocol described above, and therefore are faulty. But there are no more than t faulty processes, a contradiction.

2. If G is proper, it sends $[initial, G, m]$ messages to all the processes. At least $n - t$ processes are proper, and send $[echo, G, m]$ to all the processes. Therefore, every proper process receives $[echo, G, m]$ from at least $n - t$ distinct processes. Since $n \geq 3t + 1$, then $n - t > (n + t)/2$, and every proper process must accept m from G. □

The communication cost of such a broadcast protocol is $O(n^2)$ messages/broadcast, compared to $O(n)$ required by a "naive" broadcast (where a message is sent directly to the n destinations).

We use the broadcast protocol to execute all the broadcasts included in the agreement protocols described in this paper.

4. A Byzantine Agreement protocol for binary messages

In this section, we describe a protocol to reach Byzantine Agreement in asynchronous systems with $0-1$ initial values. This ***binary Byzantine Agreement protocol*** is shown in Figure 1.

A process G_i starts the protocol by setting the value of the variable M to its initial value M_i. This is followed by R iterations, each one consisting of two phases.

In the first phase, a process broadcasts its current value of M, and then waits to accept similar messages from $n-t$ distinct processes. When the wait is over, it stores the set of $n-t$ accepted messages in the variable *proof*, and changes M according to the number of accepted messages with value 1. The set of (signed) messages in *proof* constitutes a proof of a message value M if it shows that M was correctly set according to the protocol. For example, after the first phase, *proof* is a correct proof for the message value $M = 1$ if and only if it includes at least $n - 2t$ signed initial messages with value 1 from

```
G_i: M := M_i
    for k = 1 to k = R do
    (* Phase 1 *)
        broadcast M;
        wait to accept M-messages from n - t distinct processes;
        proof := set of accepted messages;
        count(1) := number of accepted messages with M = 1;
        if count(1) ≥ n - 2t then M := 1
            else M := 0;
    (* Phase 2 *)
        broadcast [M, proof];
        wait to accept [M, proof]-messages, with a correct proof, from n - t distinct processes;
        count(1) := number of accepted messages with M = 1;
        compute_secret(s_k);
        if (s_k = 0 and count(1) ≥ 1) or (s_k = 1 and count(1) ≥ 2t+1) then M := 1
            else M := 0;
    od
```

Figure 1. The binary Byzantine Agreement protocol for process G_i.

distinct processes: it shows why M can be set to 1 according to the protocol.

In the second phase, a process broadcasts its new value of M with the corresponding *proof*. It then waits to accept similar messages, *with a correct proof*, from $n-t$ distinct processes. When this wait is over, it computes the secret bit s_k by executing the procedure *computes_secret*(s_k). Finally, it changes M according to s_k and the number of accepted messages with value 1.

Note that every message must include the iteration and phase number in which it is sent. Furthermore, as in [Rabi83], the messages are signed, and cannot be forged. For clarity, the phase and iteration numbers, and the signatures are omitted from the description of the protocol in Figure 1.

Theorem 1:

1. All the proper processes terminate the binary Byzantine Agreement protocol (i.e., the protocol does not deadlock).
2. If the system is proper with initial value m, then every proper process terminates the protocol with $M = m$.
3. If the system is not proper, then with probability at least $1 - (1/2)^R$, every proper process terminates the protocol with the same value M.

Proof:

1. We have to show that the "wait" statements executed by proper processes always terminate. A tedious but easy proof by induction is based on the following remark. Suppose all the proper processes reach the beginning of a given phase. Each one will broadcast the message required by that phase to all the processes. By Lemma 1, every proper process will accept the message broadcast by a proper process. Therefore, every proper process accepts a message from at least $n-t$ distinct proper processes, and it terminates the "wait" statement in this phase. So, all the proper processes will reach the beginning of the next phase. The rest of the proof is left to the eager readers.

2. Suppose $n - t$ (or more) proper processes have the same value $M = m$ at the beginning of an iteration k. We show that all the proper processes will have $M = m$ at the end of this iteration. Consequently, once an agreement on a value m is reached by the proper processes, this agreement on m will hold at the end of each subsequent iteration. The proof follows.

We first claim that, in the first phase of iteration k, every proper process G has $count(1) \geq n - 2t$ if and only if $m = 1$. Note that, at the beginning of the first phase, at most t processes broadcast a value M different from m. Therefore, among the $n-t$ messages accepted by G from distinct processes, at least $n-2t$ have $M = m$, and at most t have M different from m. So, if $m = 1$ then $count(1) \geq n - 2t$, and if $m = 0$ then $count(1) \leq t$. Note that $t < n - 2t$, and the claim is proved.

From this claim, we conclude that every proper process sets $M := m$ at the end of the first phase of iteration k. We now show that, at the beginning of the second phase of the iteration, there are no correct proofs for any value M different from m.

Suppose $m = 0$. At least $n - t$ proper processes have $M = 0$ at the beginning of iteration k. A correct proof for $m' = 1$ consists of a set of at least $n - 2t$ signed messages, from distinct processors, with value 1 at the beginning of iteration k. Since $n - 2t > t$, no such set exists.

If $m = 1$, then a correct proof for $m' = 0$ consists of a set of $n - t$ messages from distinct processes such that more than $t + 1$ of them have value 0. This is also impossible.

Since there are no correct proofs for values different than m, every proper process accepts only messages with $M = m$ in the second phase of the iteration. Therefore, at the end of this second phase, every proper process sets $M := m$, independently from the value of the bit s_k.

3. Let *state*(k) = *agreement* if all the proper processes have the same value M at the beginning of iteration k, and *state*(k) = *disagreement* otherwise. We just showed that if *state*(k) = *agreement* for some $k \geq 1$, then *state*(j) = *agreement* for all j, such that $k \leq j \leq R$.

We now show that if *state*(k) = *disagreement* for some $k \geq 1$, then with probability at least 1/2 we have *state*$(k+1)$ = *agreement*. The resulting state transitions probability diagram is illustrated in Figure 2. Part 3 of the theorem immediately follows from these state transition probabilities.

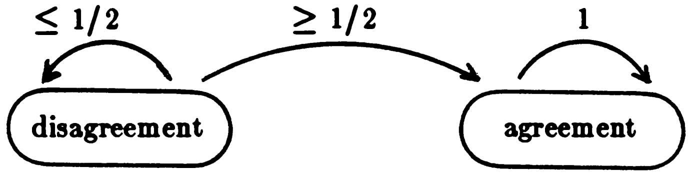

Figure 2. Arc labels denote the probability of a state transition after each iteration in the binary Byzantine Agreement protocol.

Suppose *state*(k) = *disagreement*, and consider the second phase of iteration k. Let G be the *first* proper process that accepts $n - t$ $[M, \textit{proof}]$ messages, with a correct proof, from distinct processes (say $G_1, G_2, \ldots, G_{n-t}$). There are two possible cases for G's resulting *count*(1).

(i) $count(1) \geq t + 1$. In this case, we claim that all the proper processes will have $count(1) \geq 1$ in this phase. Without loss of generality, G accepted a message with $M = 1$ from processes $G_1, G_2, \ldots, G_{t+1}$. In the same phase, every proper process G' accepts messages from all but t processes. Therefore, G' accepts a message from at least one process G_j in $\{ G_1, G_2, \ldots, G_{t+1} \}$. By Lemma 1, the messages accepted by G and G' from G_j must be identical. Therefore G' has $count(1) \geq 1$ at the end of this phase, and the claim is proved.

(ii) $count(1) < t + 1$. In this case, we claim that all the proper processes will have $count(1) < 2t + 1$ in this phase. Any proper process G' accepts messages with $M = 1$ from at most t processes in $\{ G_1, G_2, \ldots, G_{n-t} \}$, and from at most t processes in $\{ G_{n-t+1}, \ldots, G_n \}$. Therefore G' has $count(1) < 2t + 1$ at the end of this phase, and the claim is proved.

Note that (i) or (ii) is an event established before any proper process reveals its piece of the shared secret bit s_k for iteration k. Since $t + 1$ pieces are necessary to compute s_k, then the event (i) or (ii) is established before s_k is known by any process, and it is therefore independent of s_k's value. The four possible events for the second phase of iteration k are illustrated in Figure 3 with their corresponding probabilities.

	$s_k = 0$ (Prob. $= 1/2$)	$s_k = 1$ (Prob. $= 1/2$)
$count(1) \geq t + 1$ (Prob. $= \alpha_k$)	Agreement on 1 (Prob. $= 1/2\, \alpha_k$)	?
$count(1) < t + 1$ (Prob. $= 1 - \alpha_k$)	?	Agreement on 0 (Prob. $= 1/2(1 - \alpha_k)$)

Figure 3. Possible events in the second phase of iteration k, and the corresponding probabilities.

Let α_k denote the probability that $count(1) \geq t + 1$. If $s_k = 0$ and $count(1) \geq t + 1$, then all the proper processes have $count(1) \geq 1$, and therefore they all set $M := 1$ at the end of iteration k. This happens with probability $(1/2)\cdot\alpha_k$.

If $s_k = 1$ and $count(1) < t + 1$, then all the proper processes have $count(1) < 2t + 1$, and therefore they all set $M := 0$ at the end of iteration k. This happens with probability $1/2\cdot(1 - \alpha_k)$. Therefore, $state(k+1) = agreement$ with probability at least $1/2\cdot\alpha_k + 1/2\cdot(1 - \alpha_k) = 1/2$. □

5. An errorless Byzantine Agreement protocol for binary messages

We can modify the protocol described in the previous section into a protocol that ensures Byzantine Agreement without error, and reaches agreement within a small expected number of iterations that is independent of n and t (provided that $t < n/3$).

From the proof of Theorem 1, if at the end of an iteration k, a proper process has $s_k = 0$ and $count(1) \geq t + 1$, then every proper process will set $M := 1$ at the end of iteration k, i.e., the system will reach agreement on 1. Similarly, if at the end of an iteration k, a proper process has $s_k = 1$ and $count(1) < t + 1$, then the system will reach agreement on 0.

To achieve errorless termination

1. the statements

   if ($s_k = 0$ and $count(1) \geq t + 1$) **then broadcast** [*agreement reached on* 1]
   if ($s_k = 1$ and $count(1) < t + 1$) **then broadcast** [*agreement reached on* 0]

   are inserted at the end of the second phase of the binary Byzantine Agreement protocol described in Figure 1, and

2. we use the *Closefinish* procedure described by Rabin in [Rabi83]. This procedure is executed by all the processes *concurrently* with the binary Byzantine Agreement protocol. It consists of the following:

   (i) if a process receives an [*agreement reached on m*] message from a *new* process it broadcasts the message, and

   (ii) if a process receives $t + 1$ [*agreement reached on m*] messages from *distinct* processes, it sets $M := m$ and then terminates its execution of the binary Byzantine Agreement protocol.

Theorem 2: With the above modifications to the binary Byzantine Agreement protocol:

1. If a proper process stops, then all the other proper processes also stop.

2. If all the proper processes stop, then Byzantine Agreement is reached.

3. Within an expected number of 2 iterations one proper process will broadcast an [*agreement reached on m*] message. Within an expected number of 4 iterations all the proper processes will broadcast an [*agreement reached on m*] message, and when these messages are received all the proper processes stop.

Proof (sketch):

1. Part (i) of the *Closefinish* procedure guarantees that if a proper process G stops, the set of $t + 1$ messages that caused G to stop is sent to all the other proper processes, and they must also stop.

2. If a proper process G receives [*agreement reached on m*] messages (with the same m) from $t + 1$ distinct processes then at least one of them must originate from a proper process. From our previous remarks, when a proper process originates an [*agreement reached on m*] message, it has a proof that Byzantine Agreement will be reached on m. Note that G sets $M := m$ before it stops.

3. From the proof of Theorem 1, with probability 1/2 the first proper process to terminate the wait statement of the second phase of each iteration, will have a proof that agreement will be reached on some m. The added statement to the second phase of the protocol ensures that any process with such a proof sends a corresponding "agreement" message.

Theorem 1 also showed that within an expected 2 iterations an agreement is reached on some m. If $m = 1$ then $count(1) \geq 2t + 1$ for all proper processes and all the subsequent iterations. Within an expected number of two additional iterations, the secret bit will equal 0, and all the proper processes will have a proof that the agreement was reached for 1. A similar argument applies to the case where $m = 0$. □

6. A Byzantine Agreement protocol for multivalued messages

Suppose the initial messages are taken from a large universal set. To reach agreement in this case, the processes execute a simple protocol, called *filter protocol,* that reduces the number of message values in the system to (at most) two. The binary Byzantine Agreement protocol can then be used on the two remaining values. The filter protocol itself consists of two phases, and is illustrated in Figure 4.

G_i: $M := M_i$

(* Phase 1 *)

broadcast M;

wait to accept M-messages from $n - t$ *distinct processes*;

proof := *set of accepted messages*;

m := *value of M that occurred most often among accepted messages*;

$count(m)$:= *number of accepted messages with* $M = m$;

if $count(m) \geq n - 2t$ **then** $M := m$

else $M := 0$;

(* Phase 2 *)

broadcast $[M, proof]$;

wait to accept $[M, proof]$*-messages, with a correct proof, from* $n - t$ *distinct processes*;

m := *value of M that occurred most often among accepted messages*;

$count(m)$:= *number of accepted messages with* $M = m$;

if $count(m) = n - t$ **then** $M := m$

else $M := 0$;

Figure 4. The filter protocol for process G_i.

Lemma 2:

1. All the proper processes terminate the filter protocol.
2. If the system is proper with initial value m, then all the proper processes have $M = m$ when they terminate the filter protocol.
3. If a proper process has $M = m \neq 0$ at the end of the filter protocol, then all the proper processes have
   (i) $count(m) > (n-t)/2$, and
   (ii) $M = m$ or $M = 0$
   at the end of the filter protocol.

Proof:

1. At least $n-t$ proper processes broadcast the required message at the beginning of the first phase. Therefore, every proper process terminates the wait statement in the first phase. Similarly, all the proper processes terminate the wait statement in the second phase.

2. The proof is similar to the proof of Theorem 1, part 2, and will only be sketched here. Suppose the system is proper with initial value m. Every proper process will accept messages with $M = m$ from at least $n - 2t$ distinct processes. Note that $n - 2t > (n - t)/2$, so, for all the proper processes, m is the value that occurs most often among the accepted messages. Therefore, $count(m) \geq n - 2t$, and M is set to m by every proper process at the end of the first phase.

As in the proof of Theorem 1, one can show that, at the end of the first phase, there can be no correct proofs for a message with a value M different from m. So, in the second phase, every proper process will accept only messages with $M = m$. Therefore, $count(m) = n - t$ and M is set to m at the end of the filter protocol.

3. If a proper process G has $M = m \neq 0$ at the end of the filter protocol, then G has $count(m) = n - t$. So, G accepted messages with $M = m$ from a set S of $n - t$ distinct processes (during the second phase). During the second phase, any other proper process G' accepts messages from at least $n - 2t$ processes in S. By Lemma 1, the messages accepted by G and G' from processes in S must be identical, therefore they must have $M = m$. So, G' accepts at least $n - 2t$ messages with value m. Note that $n - 2t > (n - t)/2$, i.e., m is the value that occurs most in the messages accepted by G'. Therefore, G' sets M to either m or to 0. □

It is not difficult to show that if $M = m$ for some proper process at the end of the filter protocol, then m must be the initial value for at least one of the proper processes. The proof is, of course, left to the readers.

After executing the filter protocol, every proper process G follows the following rules, according to the value of its variables $count(m)$ and M.

1. If $count(m) > (n - t)/2$ for some $m \neq 0$, then G executes the binary Byzantine Agreement protocol, with "1" replaced by "m", and with M for initial message. During the execution of the protocol, G does not accept messages with $M \notin \{m, 0\}$.

2. If $count(m) \leq (n - t)/2$, then, by Lemma 2, it must be the case that all the proper processes have $M = 0$ at the end of the filter protocol. Therefore, in this case, G executes the binary Byzantine Agreement protocol with $M = 0$ for initial message, and it does not accept messages with $M \neq 0$ during this execution.

Theorem 3: The filter protocol, followed by the execution of the binary Byzantine Agreement protocol according to the given rules, achieves Byzantine Agreement with probability greater than $1 - (1/2)^R$.

Proof:

If the system is proper with the initial value m, then, by Lemma 2, all the proper processes have $M = m$, and therefore $count(m) = n - t$, at the end of the filter protocol. So, according to the given rules, all the proper processes will execute the binary Byzantine Agreement protocol with the initial value m (and "1" replaced by "m"). By Theorem 1, every correct process terminates this protocol with $M = m$.

If the system is not proper, there are two possible cases:

(i) All the proper processes have $M = 0$ at the end of the filter protocol. According to the given rules they all execute the binary Byzantine Agreement protocol with the initial value $M = 0$. By Theorem 1, they all terminate with $M = 0$.

(ii) Some proper process has $M = m \neq 0$ at the end of the filter protocol. By Lemma 2, all the proper processes have $count(m) > (n - t)/2$ and $M = m$ or $M = 0$ (at the end of the filter protocol). According to the given rules, they will all execute the binary Byzantine Agreement protocol with $M \in \{m, 0\}$, and with "1" replaced by "m". By Theorem 1, agreement is reached on $M \in \{m, 0\}$ at the end of the protocol with probability at least $1 - (1/2)^R$. □

To achieve errorless Byzantine Agreement within a small expected number of iterations, after the filter protocol, the processes should execute the errorless Byzantine Agreement protocol for binary messages that we described in a previous section, according to the given rules.

7. The upper bound on the number of faulty processes

Theorem 4: There are no Byzantine Agreement protocols for asynchronous systems where $t \geq \lfloor n/3 \rfloor$. This holds even if message authentication is assumed.

Proof (sketch):

Suppose, for contradiction, there is a Byzantine Agreement protocol P for a asynchronous systems with $n = 3t$ processes. Divide the processors in three groups, A, B and C such that each group contains t processes. Consider the following two scenarios:

1. All the processes in A and B have the initial value 0, and those in C have the initial value 1. We run the protocol P, but the processes in C immediately die. Since P must be able to overcome up to t failures, and the system is proper with value 0, then the processes in A and B must eventually agree on the value 0, say at time t_1.

2. All the processes in B and C have the initial value 1, and those in A have the initial value 0. We run the protocol P, but the processes in A immediately die. Exactly as in (1), we see that the processes in B and C must eventually agree on 1, say at time t_2.

Now we can combine scenario (1) and (2) to construct a third scenario where P leads the proper processes to disagreement, a contradiction.

3. Suppose all the processes in A have the initial value 0, and all those in C have the initial value 1. Moreover, assume the processes in B are faulty: they will always pretend to processes in A that their initial value is 0, and they always pretend to processes in C that their initial value is 1.

Suppose that the communication links between processes in A and C are very slow, and that messages between processes in these two groups take more than max (t_1, t_2). Now all the processes in A see *exactly* the same scenario as in (1) (note that because of the system asynchronicity, a process in A cannot distinguish between the death of a process in C, and a communication link that is very slow). Therefore, at time t_1, before they get any message from processes in C, all the processes in A agree on 0 as in (1). Similarly, all the processes in C see the same scenario as in (2), and they agree on 1. Therefore, P is not a Byzantine Agreement protocol, a contradiction. □

References

BenO83 M. Ben-Or, Another advantage of free choice: Completely asynchronous agreement protocol, *Proc. 2nd Symposium on the Principles of Distributed Systems,* Montreal, Canada, pp. 27-30, Aug. 1983.

Brac83 G. Bracha and S. Toueg, Resilient consensus protocols, *Proc. 2nd Symposium on the Principles of Distributed Systems,* Montreal, Canada, pp. 12-26, Aug. 1983.

Brac84 G. Bracha, An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol, *Technical Report TR 84-590,* Cornell University, Jan. 1984.

Dole81 D. Dolev, Unanimity in unknown and unreliable environment, *Proc. 22nd Annual Symposium on Foundations of Computer Science,* Nashville, Tennessee, pp. 159-168, Oct. 1981.

Dole82 D. Dolev, Polynomial protocol for multiple process agreement, *Proc. 14th Annual ACM Symposium on Theory of Computing,* San Francisco, California, pp. 404-407, May 1982.

Dole83 D. Dolev and H. R. Strong, Authenticated algorithms for Byzantine Agreement, *SIAM J. Comput.,* vol. 12, no. 4, pp. 656-666, Nov. 1983.

Fisc83 M. J. Fischer, N. A. Lynch, and M. S. Paterson, Impossibility of distributed consensus with one faulty process, *Proc. 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database systems,* Atlanta, Georgia, pp. 1-7, March 1983.

Lamp82 L. Lamport, R. Shostak, and M. Pease, The Byzantine Generals problem, *ACM Transactions on Programming Languages and Systems 4,* pp. 382-401, 1982.

Lync82 N. Lynch, M. Fischer, and R. Fowler, A simple and efficient Byzantine Generals algorithm, *Proc. Second IEEE Symposium on Reliability in Distributed Software and Data Base Systems,* Pittsburgh, Pennsylvania, pp. 46-52, 1982.

Peas80 M. Pease, R. Shostak and L. Lamport, Reaching agreement in the presence of faults, *Journal of the ACM,* vol. 27, no. 2, pp. 228-234, April 1980.

Rabi83 M. Rabin, Randomized Byzantine generals, *Proc. 24th Symposium on Foundations of Computer Science,* Tucson, Arizona, pp. 403-409, Nov. 1983.

Sham79 A. Shamir, How to share a secret, *Comm. of the ACM,* vol. 22, no. 11, pp. 612-613, Nov. 1979.