Stopping Times of Distributed Consensus Protocols: A Probabilistic Analysis*

Ozalp Babaoglu

TR 86-756

May 1986
(Revised October 1986)

Department of Computer Science
Cornell University
Ithaca, NY 14853

*Partial support for this work was provided by the National Science Foundation under Grant DCR-86-01864 and AT&T under a Foundation Grant

Stopping Times of Distributed Consensus Protocols: A Probabilistic Analysis*

Özalp Babaoğlu

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

ABSTRACT

Given a model where each processor remains correct for an exponentially distributed random time and then fails independently of the others, we characterize system executions that permit the processors to reach consensus. We show that with non-zero probability, a protocol can achieve consensus even during executions where the number of actual processors to fail exceeds its resiliency.

October 16, 1986

* Partial support for this work was provided by the National Science Foundation under Grant DCR-86-01864 and AT&T under a Foundation Grant

1. Introduction

In a distributed computing system, autonomous processors periodically need to coordinate by reaching a common decision regarding a data value or course of action. The *distributed consensus problem*† has been formulated to achieve this coordination in the presence of faulty processors [PSL80].

In a system with n processors, each processor i has an initial local value v_i drawn from a domain of at least two elements. Formally, a protocol achieves consensus if, upon its termination, each correct processor i obtains an n-element vector $\mathbf{W}_i$ such that:

IC1. If i and j are two correct processors, then they obtain the same vector (i.e, $\mathbf{W}_i = \mathbf{W}_j = \mathbf{W} = (w_1, w_2, \cdots, w_n)$).

IC2. If processor j is correct, then the jth component of the vector contains the initial local value of processor j (i.e., $w_j = v_j$ for each correct processor j).

IC3. The termination time of the protocol is bounded and known *a priori* to all processors.

Note that the protocol requirements do not specify values for the elements of the consensus vector $\mathbf{W}$ corresponding to faulty processors; it suffices for all correct processors to compute the same values. Typically, the processors apply a common deterministic function on the consensus vector $\mathbf{W}$ (such as arithmetic mean, median or majority) to obtain the common decision value.

We consider a system of n processors with authentication connected through a perfectly-reliable point-to-point communication network. In this environment, the following results are known about deterministic consensus protocols:

(i) Consensus is possible in the presence of any number of faulty processors [PSL80],

(ii) Consensus cannot be achieved in fewer than m rounds of message exchange if there can be up to $m-1$ faulty processors ‡ [DS83].

We say that a protocol is ρ-*resilient* if it can guarantee consensus in any execution where no more than ρ processors are faulty. Thus, the lower bound

---

† There are numerous formulations of the consensus problem. Collectively, they are often referred to as the *Byzantine Agreement* problem. See [Fisc83] and [SD83] for two brief surveys.

‡ Strictly speaking, this lower bound was stated for the *reliable broadcast* problem (also called the *Byzantine Generals* problem [LSP82]). The consensus and reliable broadcast problems are equivalent in the sense that given a solution to the reliable broadcast problem, consensus can be solved in the same number of rounds and given a solution to the consensus problem, reliable broadcast can be solved with one additional round [Fisc83]. Therefore, lower bound results for one problem are also lower bound results for the other.

result of (ii) could be restated as: An m-round execution can be at most $(m-1)$-resilient.

In this paper, we study the consensus problem in environments where there is no *a priori* bound on the number of processors that may be faulty during any fixed length execution. As such, it becomes impossible to bound the protocol execution time in order to guarantee consensus in an arbitrary execution. By characterizing those executions that are not "worst case" and quantifying their likelihood of occurrence (out of all possible executions), we are able derive expressions for the probability with which consensus is achieved after an arbitrary fixed length execution.

The next section defines the stochastic model for processor failure times. Sections 3 and 4 contain our results.

2. Processor Failures

There are two aspects to modeling faults in a system: we need to specify what a processor can do once it becomes faulty, and then specify the times at which it becomes faulty. The first aspect of this characterization typically takes the form of describing possible deviations from an abstract specification of correct behavior for processors. The second aspect requires a temporal description of failures during arbitrary executions. Without this characterization, we can only assume that all of the processors that are faulty by the end of an execution were already faulty initially. Previous studies of the consensus problem have been in the absence of a time sequence characterization of processor failures. The only exception to this is the work of Dwork and Moses where certain patterns of processor failures are exploited to achieve fast simultaneous Byzantine Agreement for a restricted fault model [DM86].

Each of the n processors in the system can be in two states: correct and faulty. The behavior of a processor in the *correct* state conforms to its specification encoded as a program. After some time, a processor *fails* and becomes faulty thereafter. Note that "correctness" and "faultiness" are only classifications of the internal state of a processor. A *faulty* processor may deviate from its specification in any arbitrary manner. This behavior includes choosing not to display its faultiness to the other processors until some later time as well as colluding with other faulty processors (through an adversary scheduler).

We assume that the time a processor spends in the correct state before becoming faulty is an exponentially distributed random variable with rate λ. In other words, the probability that a processor has failed by time t is given by $1-e^{-\lambda t}$. Each processor fails independently of all the others and at the same rate. The exponential distribution assumption for failure times

has been empirically justified for a large class of components, including electronic hardware, that do not "age" [SS82]. The independence of failures is typically achieved through physical and electrical isolation of the processors. We do not consider the possibility of repairing and reintegrating faulty processors into the system.

The standard "Byzantine Failure Assumption" where a system has to confront the maximum allowable number of faulty processors at the outset is unrealistic and leads to overly pessimistic designs. In real systems, the initial state contains some collection of correct and faulty processors. As the system operates, failures occur and additional processors migrate from the correct to the faulty state. Consequently, the usual malicious-processors-being-guided-by-an-adversary-scheduler assumption has to be augmented with a reasonable characterization of the time sequence of failures. Our characterization, which obliges processors to remain correct for independent and exponentially distributed random times, is one possibility. Undoubtably there are other characterizations that need to be explored [DM86].

3. Probabilistic Analysis of Consensus

Consider the execution of an m-round consensus protocol in a system with n processors where each round is α time units. We will characterize such executions by the sequence of integer-valued positive random variables $Y_1 Y_2 \cdots Y_m$ where Y_i denotes the number of processors that failed *during* round i. The random variable $N_i = \sum_{j=1}^{i} Y_j$ represents the total number of processors that have failed *by the end of* round i. Without loss of generality, we assume that all of the processors are initially correct. Recall that according to our failure model, a processor that fails during round i can choose not to exhibit its faultiness until some later round $j \geq i$, at which time it can collude with the other N_j faulty processors.

An m-round execution is $(m-1)$-resilient. Therefore, if $N_m < m$, then consensus will be achieved with certainty. Now we examine the case where there are m or more failures during an m-round execution (i.e., $N_m \geq m$).

The success of consensus in executions where the protocol resiliency is violated depends on the time sequence of the failures. Intuitively, if there are a sufficient number of failures early on in an execution, the faulty processors can choose to exhibit their faultiness at the most inopportune moments so as to prevent consensus. For example, in an m-round execution where $Y_1 = m$, consensus is impossible as dictated by the lower bound results [DS83, Hadz84]. On the other hand, if failures occur sufficiently late in the execution, there will be a reasonable chance for consensus despite the fact that resiliency may be violated. What we need to do is obtain a characterization of those executions that permit consensus for any N_m.

Definition. *(Lean Prefix):* An m-round execution *has a lean prefix*, denoted $LP(m)$, if and only if there exists some $k \leq m$ such that $N_{k-1} = N_k = k-1$.

This definition formalizes our intuition above that desired "not too many failures early on." It characterizes the first round in an execution where the number of processor that have failed is smaller than the current round number. Clearly, the definition implies that a lean prefix is always terminated by a failure-free round.

Lemma 1. In a distributed system with n processors, consensus can be achieved in an m-round execution if and only if the execution has a lean prefix.

Proof: We consider the "only if" case first.

Consider an m-round execution that does not have a lean prefix. For each round $1 \leq i \leq m$, it must be that $N_i \geq i$. In other words, at any point in the execution, there is an abundance of faulty processors. In particular, for any round in the execution, the faulty processors can behave such that a different processor exhibits faulty behavior in every round. Such executions characterize precisely the "serial faultiness" behavior that is at the heart of the lower bound proof of Dolev and Strong for the time complexity of reliable broadcast protocols with authentication [DS]. Therefore, since reliable broadcast cannot be achieved in such an execution, consensus cannot be achieved either.

Now we consider the "if" case for the lemma. We will exhibit a consensus protocol that consists of n concurrent "early stopping" (eventual) protocols that solve the reliable broadcast problem [DRS82]. In round one, each processor initiates an instance of the reliable broadcast protocol (acting as the transmitter) in order to disseminate its local input value. From then on, it also participates in the $n-1$ reliable broadcast protocols initiated by the other processors. Given this construction, the success of a protocol to achieve consensus in m rounds depends on the length of the longest reliable broadcast protocol—if all n protocols terminate by round m then consensus is guaranteed. Note that a processor may decide on values corresponding to each of the concurrent reliable broadcasts in different rounds. Once a processor decides on a value corresponding to a particular broadcast, it stops participating in that protocol by not sending any messages and ignoring all incoming messages. However, it continues to participate in the protocols corresponding to undecided values.

By the hypothesis, the m-round composite execution has a lean prefix. Let f be the length of the lean prefix for this execution and let h denote the total number of processors that fail during the m rounds. By definition, $N_{f-1}=f-1$, round f is failure free and $h-f+1$ additional failures occur in

rounds $f+1$ through m. Consider the behavior of the N_{f-1} faulty processors that are present by the end of round f. Regardless of when these processors failed or how they exhibited their faultiness, each of the n early stopping reliable broadcast protocols encounters a f-round execution where at most $f-1$ processors fail and the last round is failure-free. This is precisely the characterization for the termination of an early stopping protocol† [DRS82, Hadz84]. Of course, processors may have decided on some of the values at rounds earlier than f; round f represents the latest time by which any of the n reliable broadcasts can terminate.

By the end of round f, all correct processors are able to construct the consensus vector **W** from the n decision values. During rounds $f+1$ through m the correct processors remain silent and ignore any messages they may receive from other (faulty) processors. Thus waiting until round m serves to synchronize the correct processors with respect to the termination of the consensus protocol. Finally, note that the vector **W** constructed by the correct processors at round $f+1$ continues to satisfy the requirements for consensus at round m. The $h-f+1$ entries corresponding to the processors that fail during rounds $f+1$ through m are identical in all of the consensus vectors of correct processors. Thus they continue to satisfy the requirements for consensus by the end of round m.

□

Next we derive the probability with which an m-round execution achieves consensus. The following result will be used repeatedly in our derivations.

Lemma 2. Given a system with n processors that are all correct initially, the probability that i of them are faulty after t time units is given by

$$\varphi(n, i, t) = \binom{n}{i}(1-e^{-\lambda t})^i(e^{-\lambda t})^{n-i}, \quad 0 \le i \le n.$$

Furthermore, the number of processors that fail during disjoint intervals have a common distribution.

Proof: Since processors fail independently, we can interpret the failure of each of them during t time units as a "success" outcome in n independent trials. The probability of a success is simply the probability that an exponential random variable with rate λ is smaller than t. The desired probability expression is the binomial distribution [Fell71]. The common distribution result is a consequence of the "memoryless" property of the exponential distribution.

---

† More precisely, the stopping time of an early stopping protocol is $f+2$ rounds given that the execution encountered f failures. By requiring only $f+1$ rounds to stop, we are being slightly optimistic in our analysis.

□

Now we can derive the expression that quantifies the probability that an arbitrary execution has a lean prefix.

Lemma 3. In a system with n processors, an m-round consensus protocol execution during which f processors fail has a lean prefix with probability

$$\Lambda(m,f) = \mathbf{P}\{LP(m) \mid N_m = f\}$$

$$= \sum_{k=1}^{m} \sum_{\vec{Y} \in \mathbf{Y}} \frac{f!}{y_1! y_2! \cdots y_{k-1}! (f-k+1)!} \left[\frac{1-e^{-\lambda\alpha(m-k)}}{1-e^{-\lambda\alpha m}}\right]^f \left[\frac{1-e^{-\lambda\alpha}}{1-e^{-\lambda\alpha(m-k)}}\right]^{k-1} \frac{e^{-\lambda\alpha k f}}{e^{-\lambda\alpha(k y_1 + (k-1)y_2 + \cdots + 2y_{k-1})}}$$

where

$\vec{Y} = (Y_1 = y_1, Y_2 = y_2, \cdots, Y_m = y_m)$ and

$\mathbf{Y} = \{\vec{Y}\colon\ N_{k-1} = k-1,\ y_k = 0,\ N_m = f\}$

is the set of all m-round executions with f failures and a length k lean prefix.

Proof: By definition, an m-round execution $Y_1 Y_2 \cdots Y_m$ where f processors fail has a lean prefix of length k if and only if

$$N_{k-1} = k-1,$$

$$Y_k = 0 \text{ and}$$

$$N_m - N_{k-1} = f-k+1.$$

The joint distribution of these events can be obtained easily by conditioning on the number of failures in each round of execution (starting with $Y_1 = y_1$) and repeatedly applying Lemma 2. Given the probability of the execution having a lean prefix of length k, the result follows by summing over all $k \leq m$.

□

In order to gain insight into the magnitude of this expression, we have computed $\Lambda(m,f)$ as a function of f for various m and α in a system with 13 processors. In the results displayed in Figure 1, we have set $\lambda = 1$. Consequently, the unit of time in this system coincides with the mean-to-failure of a single processor ($1/\lambda$). Obviously, for each execution, the probability of a lean prefix when $f < m$ is one. Note that in an m-round execution, there is a significant probability of a lean prefix (and, thus consensus) existing even for

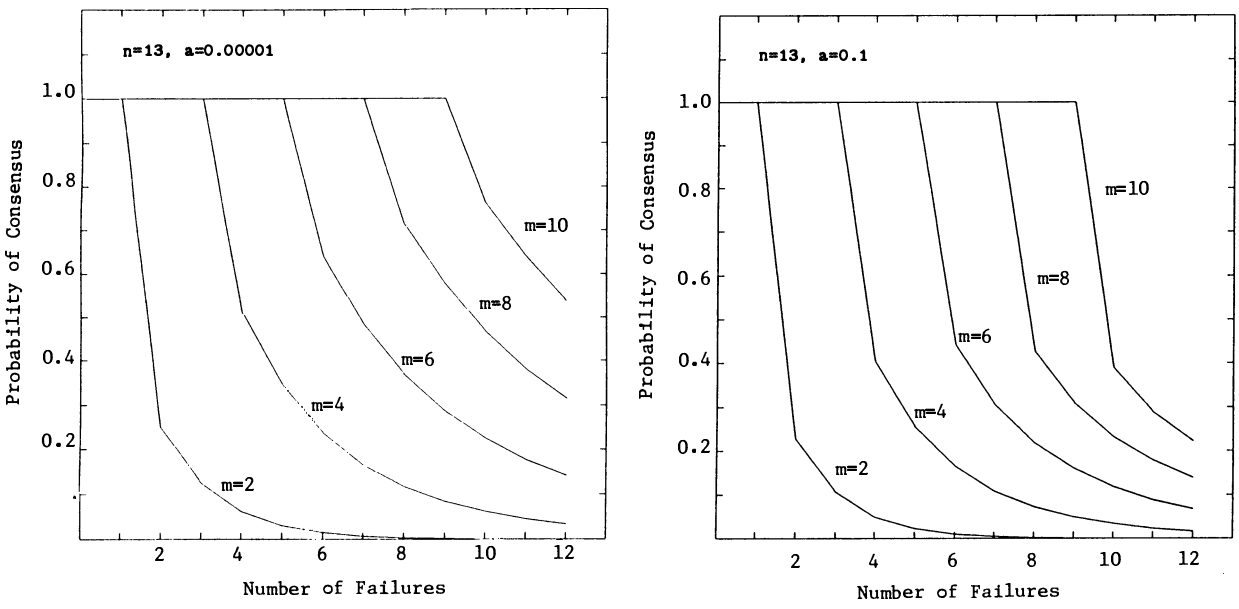

Fig. 1. Probability that an m-round execution has a lean prefix.

f as large as $2m$. The probability of a lean prefix falls off more rapidly with increasing f when the system has a longer round length.

We note that the distribution of the failure times of processors during the m rounds approaches the uniform distribution as $\lambda\alpha \to 0$. In other words, as the round length becomes shorter, the probability of any one of the m rounds containing the failure time of a processor approaches $1/m$. This is confirmed by the observation that $\lim_{\lambda\alpha\to 0} \Lambda(m,f)$ approaches the multinomial distribution [Fell71] where $\mathbf{P}\{Y_i = y_i\} \to (1/m)^{y_i}$ for all $1 \le i \le m$.

Theorem 1. In a system with n processors, an m-round protocol achieves consensus with probability

$$\Gamma(m) = \sum_{i=0}^{m-1} \varphi(n, i, \alpha m) + \sum_{f=m}^{n-1} \Lambda(m, f)\varphi(n, f, \alpha m).$$

Proof: By Lemma 1, the probability of consensus in an m-round execution is equal to the probability that the execution has a lean prefix

$$\Gamma(m) = \mathbf{P}\{LP(m)\}.$$

Conditioning on the number of failures during the execution, we obtain

$$\mathbf{P}\{LP(m)\} = \mathbf{P}\{N_m < m\} + \sum_{f=m}^{n-1} \mathbf{P}\{LP(m) \mid N_m = f\}\mathbf{P}\{N_m = f\}$$

since the probability of a lean prefix in m rounds, given $N_m < m$, is one. The result follows after substituting the expressions of Lemmas 2 and 3.

□

4. Asymptotic Results

The results of the previous section could be interpreted as answering the question "what is the probability that the vector constructed after an m-round execution satisfies the requirements for consensus?" In this section, we consider the dual question "how long does the protocol have to execute to achieve consensus?" By Lemma 1, we know that this equivalent to asking "how long does the protocol have to execute until it encounters a lean prefix?" We will be able to answer this question in the limit as $n \to \infty$.

Given that each processor fails at rate λ, the composite failure process of a system with n processors initially has rate $n\lambda$. After the ith failure, this rate decreases to $(n-i)\lambda$. Consider the behavior of a system as $n \to \infty$. To keep the composite failure rate finite, we will scale the individual processor failure rates by $1/n$. The events corresponding to processors failing in this system constitute a Poisson process with rate λ. As before, we characterize an execution through the random variables $Y_1 Y_2 Y_3 \cdots$ Unlike before, we consider executions of arbitrary length. Given that the failures constitute a Poisson process, Y_i are independent and identically distributed as Poisson random variables with mean $\lambda\alpha$. Let S be an integer-valued positive random variable such that $S = s$ if and only if round s terminates a lean prefix of the execution. In other words,

$$S = \min\{s: Y_1 + Y_2 + \cdots + Y_s = s - 1\}.$$

Define the random variable $X_i = 1 - Y_i$. Rewriting the above expression in terms of X_is, we obtain

$$S = \min\{s: X_1 + X_2 + \cdots + X_s = 1\}.$$

Since the event $S = s$ is independent of $X_{s+1} X_{s+2} \cdots$, the random variable S is called a *stopping time* for the sequence $X_1 X_2 X_3 \cdots$ [Ross70]. We are interested in $\mathbf{E}[S]$. Since S is a stopping time, Wald's equation [Ross70] allows us to write

$$\mathbf{E} \sum_{i=1}^{S} X_i = \mathbf{E}[S]\mathbf{E}[X].$$

By definition, the left hand side is equal to 1. Also, $\mathbf{E}[X] = 1 - \mathbf{E}[Y] = 1 - \lambda\alpha$. Substituting and solving for $\mathbf{E}[S]$, we obtain

$$\mathbf{E}[S] = \frac{1}{1-\lambda\alpha}.$$

Thus, we expect the protocol to run for $\mathbf{E}[S]$ rounds before a lean prefix is encountered. Note that as $\lambda\alpha \to 1$, the expected stopping time grows unbounded and the protocol will never be able to find a lean prefix no matter how long it executes.

5. Conclusions

We have argued that a meaningful fault model for reliable distributed computing must include a time sequence characterization of failures in addition to the usual faulty behavior characterization. We presented one such characterization that was motivated by empirical failure data on processor lifetimes. Using this model, we identified those executions for which consensus can be achieved even though the resiliency of the protocol is violated. Our results indicate that these executions occur with non-negligible probability amongst the set of all possible executions. Thus, an m-round execution may be able to achieve consensus in the presence of many more than $m-1$ faulty processors.

Acknowledgments

I am grateful to Pat Stephenson for many discussions of this material and also for his help in generating the numerical data.

References

DM86 C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine Environment I: Crash failures, Private Communication (January 1986).

DRS82 D. Dolev, R Reischuck, and H. R. Strong, 'Eventual' is earlier than 'immediate', *Proc. 23rd Symposium on Foundations of Computer Science*, Chicago, Illinois, pp. 196-203, November 1982.

DS83 D. Dolev and H. R. Strong, Authenticated algorithms for Byzantine Agreement, *SIAM J. Comput.*, vol. 12, no. 4, pp. 656-666,

November 1983.

Fell71 W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, New York (1971).

Fisc83 M. J. Fischer, The consensus problem in unreliable distributed systems (A Brief Survey), Tech. Rep. YALEU-DCS-RR-273, Dept. of Computer Science, Yale University, New Haven, Connecticut, June 1983.

Hadz84 V. Hadzilacos, Issues of fault tolerance in concurrent computations. Ph.D Thesis, Tech. Rep. TR-11-84, Aiken Computation Laboratory, Harvard University, Cambridge, Mass., June, 1984.

LSP82 L. Lamport, R. Shostak, and M. Pease, The Byzantine Generals problem, *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, July 1982.

PSL80 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.

Ross70 S. M. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, California, 1970.

SD83 H.R. Strong and D. Dolev, Byzantine agreement. In *Digest of Papers, Spring Compcon 83*, San Francisco, California, pp. 77-81, March 1983.

SS82 D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, Belford, Mass. (1982).