Efficient agreement using fault diagnosis*

Brian A. Coan

Bellcore, 445 South Street, Morristown, NJ 07960, USA

Received April 1990 / Accepted July 1993

Summary. We give an extremely simple Byzantine agreement protocol that uses $O(t^2)$ processors, $\min(f + 2, t + 1)$ rounds of communication, $O(n \cdot t \cdot f \cdot \log |V|)$ total message bits, and $O(\log |V|)$ maximum message size, where $n$ is the total number of processors that actually participate in the protocol, $t$ is an upper bound on the number of faulty processors, $f$ is the number of processors that actually fail in a given execution, and $V$ is the set of possible inputs. This protocol uses roughly the same resources as a more complex protocol due to Dolev, Reischuk, and Strong. By adding explicit fault diagnosis to our first protocol, we produce a somewhat more complicated protocol that uses $O(t^{1.5})$ processors, $\min(f + 2, t + 1)$ rounds, $O(n \cdot t^2 \cdot f \cdot \log |V|)$ total message bits, and $O(t \cdot \log |V|)$ maximum message size.

Key words: Byzantine agreement – Fault-tolerant protocols – Fault diagnosis – Deterministic synchronous protocols

1 Introduction

Byzantine agreement is a fundamental problem that was introduced by Pease, Shostak, and Lamport [19]. Over the past decade, many variants of the problem have been studied. In this paper, we consider a system of deterministic processors that communicate by message passing in a series of synchronous rounds. Some processors are correct, following their protocol exactly. The rest experience Byzantine faults, deviating from their protocol in arbitrary ways. Each processor begins with an input and eventually each correct processor must irrevocably decide on an answer. The answers of the correct processors must satisfy two conditions: (1) all the answers are the same, and (2) if all the inputs are the same, then all the answers must equal the common input. For this version of the problem, there is currently a gap between the known lower bounds on resources and the most efficient known protocols. The results in this paper help narrow that gap. Specifically we reduce the number of processors needed by an agreement protocol that uses small messages and terminates in an optimal number of rounds.

We are interested in minimizing the use of the following four resources: processors, rounds of communication, total number of message bits, and size of the largest message. If $n$ is the total number of processors, $t$ is an upper bound on the number of faulty processors, and $f$ is the number of processors that actually fail in a given execution, then the following lower bounds are known: $3t + 1$ processors [14, 19], $\min(f + 2, t + 1)$ rounds [12], $O(n \cdot t)$ total message bits [11], and $O(1)$ message size. At present there is a trade-off in the use of these resources. Although each of the four lower bounds is known to be achievable, there is no single protocol that is optimal in its use of all four resources. Much recent work on Byzantine agreement has been a search for an improvement in the trade-off.

The lower bounds are achievable independently. For example, the original protocol of Lamport, Shostak, and Pease [17] achieves the bounds on processors and rounds but requires a maximum message size, and hence also a
total number of message bits, that is exponential in \( t \). A protocol of Bar-Noy and Dolev [1] and the first protocol of this paper (with the size of the input set fixed at 2) achieve the lower bounds on rounds and message size but exceed the lower bounds on processors and total message bits. The protocols of Berman, Garay, and Perry [4] and Coan and Welch [7] achieve the lower bounds on processors and total message bits but exceed the lower bounds on rounds and message size.

The worst-case lower bound on rounds is \( t + 1 \) [13]. Dolev, Reschuk, and Strong [12] pointed out that there are protocols that terminate faster in those executions where the actual number of faulty processors is small. They gave the name early stopping to this property. Our protocols have early stopping. In an execution where processors actually fail, our protocols terminate in \( (f + 2, t + 1) \) rounds, which is optimal [12].

In this paper we give two new protocols. The first protocol uses \( O(t^2) \) processors, \( (f + 2, t + 1) \) rounds of communication, \( O(n \cdot t \cdot f \cdot \log |V|) \) total message bits, and \( O(\log |V|) \) maximum message size, where \( n \) is the total number of processors that actually participate in the protocol, \( t \) is an upper bound on the number of faulty processors, \( f \) is the number of processors that actually fail in a given execution, and \( V \) is the set of possible inputs. The second protocol uses \( O(t^{1.5}) \) processors, \( (f + 2, t + 1) \) rounds, \( O(n \cdot t^2 \cdot f \cdot \log |V|) \) total message bits, and \( O(t \cdot \log |V|) \) maximum message size.

Our first protocol, our second protocol, our objective is to reduce the number of processors by "reusing" groups (i.e., by having the same group play a special role in more than one round). In most rounds we do explicit fault diagnosis on all of the processors in the special group for the previous round. Often this fault diagnosis allows each correct processor to correctly identify a faulty processor. A correct processor ignores certain messages from processors that it has diagnosed as faulty. After using a group as the special group and doing explicit fault diagnosis on the group, we can productively reuse the group as a special group in a subsequent round. The answer of the correct processors will be "locked in" by the first round in which the special group contains no undiagnosed faulty processors; this happens by round \( t + 1 \).

Explicit and implicit fault diagnosis are fundamentally different techniques. Explicit fault diagnosis entails the exchange of messages to gather evidence of specific erroneous behavior by specific processors. It results in each correct processor constructing a set of the processors that it has diagnosed as faulty. Future messages from these diagnosed processors can be ignored safely. Implicit fault diagnosis requires neither the exchange of additional messages nor the collection of evidence of specific erroneous behavior by specific processors. Rather, it entails an inference that because a protocol execution has reached a specific round without a decision being reached there must be a collection of disjoint sets of processors, each with a non-zero lower bound on the number of faulty processors it contains. The identity of the collection of disjoint sets is a function of the current round number.