Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement

Russell Turpin[1]
Information Research Associates
911 West 29th Street
Austin, TX 78705

Brian A. Coan[2]
MIT Laboratory for Computer Science
Cambridge, MA 02139

April 1984

Abstract

A binary Byzantine agreement algorithm can be extended to produce a multivalued Byzantine agreement algorithm. The resulting multivalued algorithm is cheaper than previously published algorithms when the cost of transmitting values from the multivalued domain is significant.

Keywords

Byzantine generals, Byzantine agreement, fault tolerance in distributed systems.

---

[1] This author eschews government funding.

[2] This author's work was supported in part by the Office of Naval Research under contract N00014-82-K-0154, the Office of Army Research under contract DAAG29-79-C-0155, and the NSF, under grants MCS-8116678, MCS-8302391, and MCS-8306854.

Introduction

The concern of this paper is a set of potentially faulty processes that engage in a distributed computation to agree on some piece of information. Each process enters the computation with an initial value. The computation returns a common result value to each correct process. If all correct processes begin the computation with identical initial values, then the result value equals the initial value.

The computation can be briefly characterized as follows. The computation is fully distributed and symmetric. It includes several rounds of synchronous message exchange over a completely connected, totally reliable communications network. The correct processes communicate only through messages. The communications network correctly identifies the sender of each message to the recipient of the message. Processes are assumed to have no signature ability (authentication). That is, there is no immediate way of detecting whether or not a relayed message has been altered.

A process fails if it does not successfully perform the actions prescribed by the agreement algorithm. No assumptions are made restricting the messages sent by faulty processes. One can imagine that all faulty processes act maliciously, in collusion, and with magical knowledge of the state of the distributed system.

A computation that functions as described above solves the Byzantine generals problem without authentication [3]. (Authenticated protocols protect relayed messages from alteration.) Let P be the number of processes that engage in the agreement computation and let T be an upper bound on the number of processes that may fail during the agreement computation. Byzantine agreement without authentication requires $P>3T$ [6], and cannot be achieved in fewer than $T+1$ rounds [4].

A less general formulation of the problem assumes that a distinguished process transmits initial values to the other processes. This paper makes no assumption about the source of the processes's initial values.

This paper describes a method for extending a binary Byzantine agreement algorithm to reach agreement on values from an arbitrary domain V. Any binary algorithm that does not require a distinguished transmitter process may be used. Two rounds are prepended to the binary algorithm. In the first round, each process sends every other process its initial value. In the second round, each process broadcasts a single bit of information by sending or not sending null messages. The third and subsequent rounds follow the chosen binary algorithm.

Previous algorithms for reaching Byzantine agreement on values from an arbitrary domain V require processes to send messages whose length depends on the size of V in each round of the computation. Using the extension described in this paper, messages whose length depends on the size of V are sent only in the first round. Since the time that must be allotted each round of the computation depends in part on the length of messages sent in the round, the extension enables significant savings when the domain is large.

The prepended rounds are an integral part of the extended computation. In particular, agreement can be guaranteed only if no more than T processes fail during the computation, including the first two rounds, where $P>3T$. (The chosen binary algorithm may make additional assumptions.)

The body of this paper contains three sections: a description of the extension, a proof of its correctness, and a discussion of implementation concerns and performance characteristics.

Description of the Extension

In the first round, each process sends its initial value to every other process. A process is said to be *perplexed* if, in the first round, it receives at least as many as $(P-T)/2$ initial values different from its own. Processes that are not perplexed are said to be *content*. In the second round, each perplexed process sends a message to every other process. The semantics of this message is just "I am perplexed".

Each process maintains three local variables: two arrays indexed by process number and a boolean. These variables are are assigned values during the first two rounds. For process j, and $i \neq j$,

these variables are defined as follows:

v(j) The process's initial value.

v(i) The initial value received from process i.

p(j) A boolean that is set true if and only if process j is perplexed, that is, $v(j) \neq v(i)$ for at least as many as $(P - T)/2$ distinct values of i.

p(i) A boolean that is set true if and only if process i sent a message claiming it is perplexed.

alert A boolean that is set true if and only if at least as many as $P - 2T$ elements of p are true.

The binary computation is used to reach agreement on alert. If the binary computation agrees alert = true, there are correct processes with different initial values from V. In this case, all correct processes use a predefined default value from V as the result of the extended computation. If agreement is alert = false, then all correct content processes have the same initial value from V. This value is the result of the extended computation. Perplexed processes deduce this result by using the initial value that is common to a majority of the content processes. That is, each perplexed process tabulates as votes the values v(j) for which p(j) is false. The majority vote is for the value favored by the correct content processes.

Proof of Correctness

The extended computation is correct if (1) all correct processes obtain the same result value, and (2) the result value equals the common initial value whenever all correct processes begin with the same initial value.

The second claim is easily proved. If all correct processes have the same initial value from the domain V, then no correct process is perplexed and all correct processes have alert = false. The binary computation agrees alert = false and all correct processes, which are content, use their initial value as the result.

The first claim has two cases: the binary computation agrees alert = true or alert = false. In the

former case, all correct processes select the default value as the result of the extended computation. In the latter case, it is necessary to show that all content processes have the same initial value and that this value is deduced by all the perplexed processes. This will now be demonstrated.

Any subset of more than $(P+T)/2$ processes contains a majority of the correct processes. From this basic fact, it follows that each content process has the same initial value as a majority of the correct processes. (Observe that $(P+T)/2$ and $(P-T)/2$ sum to P.) Since there cannot be two distinct majorities, all content processes have the same initial value.

Since the result of the binary computation is alert = false, there are at least $T+1$ correct content processes, for otherwise there would be at least $P-2T$ correct perplexed processes and all correct processes would be alert and the result of the binary computation would be alert = true. Each perplexed process has p(j) false for all content processes and possibly for some incorrect processes. Since there are at most T incorrect processes, the content processes are a majority of those for which p(j) is false. Taking a majority vote of the v(j) for which p(j) is false produces the value shared by the content processes.

Implementation and Performance Analysis

Many binary algorithms favor one of the two values in the binary domain. The binary algorithms (without authentication) described in [1,2,3,5] all reach agreement for the favored value whenever more than T correct processes begin with that value. (Assume that the threshold LOW equals $T+1$ in [1,2,5].)

In the extended algorithm, the second round together with the binary computation can be interpreted as reaching binary agreement on which processes are perplexed, providing agreement is reached for perplexed whenever $(P-T)/2$ or more correct processes are initially perplexed. If the chosen binary algorithm exhibits the bias described above, the second round of the extended algorithm can be omitted. (The chosen binary algorithm must require that each process sends all other processes initial binary values so that the values in the array p can be set.)

A good multivalued Byzantine agreement algorithm is presented in [5]. Agreement is reached in $2T + 4$ rounds and requires $O(P^3)$ messages each comprising $O(\log P \log |V|)$ bits. The extension described in this paper using the algorithm in [5] to reach binary agreement reaches multivalued agreement in $2T + 5$ rounds (the second round of the extension is not needed) and requires $O(P^3)$ messages having $O(\log P)$ bits and $O(P^2)$ messages having $O(\log |V|)$ bits. The latter messages are sent only in the first round.

The above analysis shows that the extension of the binary algorithm in [5] yields a multivalued algorithm that is cheaper in message bits than the multivalued algorithm described in [5]. The extension enables this savings because only in the first round does it send messages whose length depends on the size of the value domain. The actual time savings possible depends on a variety of factors, including the cost of an additional communication round relative to the cost of sending large messages, the size of the value domain, and the bandwidth of the communications network.

Conclusion

This paper shows that reaching Byzantine agreement on values from an arbitrary domain is not essentially more difficult than reaching binary Byzantine agreement, except for the necessity of initially exchanging and comparing values. Using the technique described in this paper to extend a good binary algorithm yields a multivalued algorithm faster than those previously published when agreement must be reached on large sets of data.

Acknowledgments

Mani Chandy, while teaching a distributed algorithms class, first posed the Byzantine generals problem to Russell Turpin, who also thanks Jay Misra and Doug Neuse for their criticism and advice, and J. C. Browne and the employees of Information Research Associates for their support. Brian Coan thanks Nancy Lynch, Jennifer Lundelius, and Eugene Stark for helpful discussions and suggestions. Both authors are indebted to David Gries for editorial advice.

References

[1] D. Dolev, M. Fischer, R. Fowler, N. Lynch, and H. R. Strong

An Efficient Byzantine Agreement without Authentication

IBM Research Report RJ3428, March 82,

Watson Research Center Distribution Services, P O Box 218, Yorktown Hts., NY 10598

[2] D. Dolev and H. R. Strong

Polynomial Algorithms for Multiple Processor Agreement

14th ACM Symp. on Theory of Computing (May 82) 401–407

[3] L. Lamport, R. Shostak, and M. Pease

The Byzantine Generals Problem

ACM Transactions on Programming Languages and Systems, 4 (3) (1982) 382–401

[4] N. Lynch and M. Fischer

A Lower Bound for the Time to Assure Interactive Consistency

Information Processing Letters, 14 (4) (1982) 183–186

[5] N. Lynch, M. Fischer, and R. Fowler

A Simple and Efficient Byzantine Generals Algorithm

2nd Symp. on Reliability in Distributed Software and Database Systems (1982) 46–52

[6] M. Pease, R. Shostak, and L. Lamport

Reaching Agreement in the Presence of Faults

J. ACM 27 (2) (1980) 228–234