The Best of Both Worlds: Guaranteeing Termination in Fast Randomized Byzantine Agreement Protocols

Oded Goldreich
Erez Petrank

Department of Computer Science
Technion
Haifa, Israel.

Abstract

All known fast randomized Byzantine Agreement (BA) protocols have (rare) infinite runs. We present a method of combining randomized BA protocols of a certain class with any deterministic BA protocol to obtain a randomized protocol which preserves the expected average complexity of the randomized protocol while guaranteeing termination in all runs. In particular, we obtain a randomized BA protocol with constant expected time which always terminate within $t + O(\log t)$ rounds, where $t = O(n)$ is the number of faulty processors.

Comment: Reproduced from an old troff file. Paper has appeared in *IPL*, Vol. 36, October 1990, pp. 45–49.

Keywords: Byzantine Agreement , Randomized Algorithms , Distributed Computing.

1 Introduction

Byzantine Agreement (BA) is a fault tolerant distributed task with the following input-output relations:

1. *Agreement:* For every set of inputs and any behavior of the faulty processors, all good processors should output the same value.
2. *Validity:* If all good processors start with the same input value, they should all output this value, no matter what the faulty processors do.

BA was introduced in the seminal paper of Pease, Shostak, and Lamport [PSL]. It has since been recognized as a key problem in the area and attracted much research. The status of BA, as well as other distributed tasks, heavily depends on the underlying communication model. In this note we assume that communication takes place in synchronous *rounds* and via *private* channels connecting each pair of processors. (Brief remarks on the asynchronous model can be found in section 6).

Let n be the number of processors and t be the number of faulty processors, [PSL] showed that no solution exists if $n \leq 3t$. They also gave a deterministic algorithm solving the problem for the case n $3t$. Their protocol consisted of $t+1$ rounds but required exponential amount of communication. A deterministic algorithm with polynomial message (and computation) complexity and $t+1$ rounds for $t = O(n)$ was recently introduced by Moses and Waatz[MW]. Fischer and Lynch [FL] proved that $t+1$ rounds are optimal (i.e. every deterministic BA protocol has a run which take at least $t+1$ rounds). Improvement is possible through randomization.[1] In particular, Feldman and Micali [FM], improving over previous works of Ben-Or [Be], Rabin [R], Chor and Coan [CC], and Bracha [Br], presented a *randomized* BA protocol with a constant expected time for $t = O(n)$. However, their protocol has a shortcoming: there is no bound on the number of rounds in the worst case (i.e. for the worst sequence of coin-tosses). Namely, for every m, the probability that the protocol proceeds more then m rounds is (very small, yet) greater then 0. This disadvantage is shared by all randomized BA protocols (with expected number of rounds less than $t+1$) known so far.[2] *Is this disadvantage unavoidable? Namely, must every randomized BA protocol with expected number of rounds smaller then t+1, have unbounded runs?* This question is the focus of this note.

Our main result is a randomized BA with a constant expected number of rounds in which all runs take at most $t + O(\log\ t)$ rounds. In fact we show how every randomized BA protocol of certain natural class (to which almost all known randomized BA protocols belong) can be transformed into a randomized BA which always terminates, yet has the same (up to an additive constant) expected running time as the original protocol. Hence we obtain the best of both worlds: a randomized BA protocol with expected running time as the best possible randomized BA protocol and with no infinite runs (as good deterministic BA). In other words, our randomized BA protocol is almost optimal[3] *on the average* and *on the worst case* simultaneously.

[1] Interestingly, Randomized protocols offer no advantage in case one requires concurrent termination [DM,MT].

[2] Also in the weaker model of fail-stop failures, fast *randomized* agreement protocols (e.g. [CMS]) have (rare) infinite runs.

[3] Optimality on the average means up to an additive constant, while in the worst case optimality means up to an additive logarithmic factor.

It should be stressed that naive attempts to solve the problem fail. In particular, one cannot just run a deterministic protocol in case the randomized protocol has not stopped within a predetermined number of rounds, since the problem of determining whether the protocol has stopped within a given number of rounds is as hard as the original problem (of BA). The same problem occurs when one tries to run deterministic and randomized protocols in parallel, and take the output value of the protocol that finishes earlier.

2 The Model

We consider a synchronous system of n processors with unique identities, in which every pair of processors is connected via a private line. Each processor has a local input value, and has to decide upon a local output value. We distinct between the notion of *writing a local output* and *the local termination* of a processor. When a processor has gained enough information to decide upon his output, he writes it into his output tape. In most previous works, a processor that wrote his output, finished his role in the protocol[4] We shall make each processor proceed after writing his output, for the benefit of the other processors. Therefore local termination will occur **after** the round in which the output was written. However, further participation might have been risky in the sense that further writing might be erroneous. Therefore we stress, that a processor writes an output **only once.** Once an output was written, the processor never tries to "change his mind".

The *local running time* of a processor is the number of rounds needed until its local termination. The *running time* of the protocol (for a specific input, specific behavior of the faulty processors and a given sequence of coin-tosses) is the maximum over the local running times of the good processors. The *average running time* of a protocol is defined by taking the worst inputs and the worst behavior of the adversaries, and applying the expectation operator over the coin-tosses probability space. The *worst running time* is the running time of the protocol for the worst inputs the worst behavior of the adversaries and for the worst sequence coin-tosses. In (good) deterministic protocols the worst (and average) running times are equal to $t+1$. In currently known randomized protocols the average time is less then $t+1$ (in particular [FM] has an $O(1)$ expected running time) and the worst case running time is unbounded.

3 The Protocol:

All the known randomized protocols share a structure based on the notion of *phases* (consisting of a fixed number of rounds), and the notion of *current value* (which is the "favorite value" of the processor at the end of a phase). Most of these protocols (in particular the [FM] protocol) meet the following properties:

P1. If a good processor *writes* output v during a phase, then at the end of that *phase* all good processors have v as their *current value.*

P2. If all good processors enter a *phase* with the same *current value,* then this value will be written on their output tape after that phase. (This value will also remain their current

[4] Some of the protocols ignore this question completely, thus, implicitly identifying decision with termination.

value).

Note that in most of the original protocols, writing the output is (implicitly or explicitly) identified with local termination. We shall mention explicitly stopping, and writing the output will not imply local termination. Our protocol can use any randomized protocol that meet both properties P1 and P2, and any deterministic BA protocol.

Our Protocol (for a processor P)

The protocol will be parametrized by an integer ($k \geq 1$).

1. Run up to k phases of the randomized protocol. Proceed to step 2 if local writing has taken place or the k phases are over.
2. If you wrote your output in phase $l \quad k$, then stop at the end of phase $l + 1$ (without any further writing).
3. If you wrote your output at phase k, participate in a deterministic protocol that starts after the kth phase. (Without any further writing!). Use your current value as an input to the deterministic protocol.
4. If you have not written your value during these k phases, participate in the deterministic protocol starting after the kth phase. and write the local output determined by this execution. the input to the deterministic protocol is the current value at the end of phase k.

4 Correctness

Validity: If all good processors start with the same value then by P2, during the first phase they will all write their input values.

Agreement: P1 and P2 together guarantee that, in the original randomized BA protocol, all the good processors write their output values in the same phase or during two consecutive phases. In analysing our protocol there are three possible cases to consider :

a: The first good processor to write an output, did so in phase $l \quad k$. In this case, all good processors wrote their values in phases l , $l+1$ (both smaller or equal to k). In these phases our protocol is still following the instructions of the randomized protocol, and therefore agreement is guarantied by the agreement property of the randomized protocol.

b: The first good processor to write an output, did so during the kth phase. If all the good processors wrote their value during the kth phase, then we use the same argument as in case (a). Otherwise, some of the good processors wrote their values during the kth phase, while the others entered the deterministic protocol without having written an output. By property P1, all good processors have at this stage the same current value and those which wrote an output, wrote the same output value. Remember that though some of the good processors have written their values, they all participate in the deterministic protocol. The validity property of the deterministic protocol assures us that all the remaining good processors will write their identical current value into their output tape.

c: None of the good processors has written an output during the first k phases. In this case, all the good processors write their values during the deterministic protocol, and therefore the agreement follows from the agreement property of the deterministic protocol.

We emphasize again that a processor does not write an output twice. Therefore any further participation in the protocol after writing a local output, does not change the value written before.

5 Rounds Complexity Analysis

Let r denote the number of rounds in a phase, P_k - the probability that the randomized protocol enters the kth phase, (i.e., has not finished in $k-1$ phases), T_R - the expected running time of the randomized protocol, and T_D the running time of the deterministic protocol. (All time units are in rounds). Clearly $T_R \geq r$.

Let X be the random variable representing the running time of our protocol, and let Y be the random variable representing the running time of the randomized protocol used. Let Z be the random variable that represents the additional time due to the use of the deterministic protocol. Namely, $Z = 0$ if $Y \quad k \cdot r$ and $Z = T_D$ otherwise. Obviously, $E(X) \leq E(Y + r + Z)$ (The additional term r represents the extra phase we add to the the randomized protocol). Therefore:

$$\begin{aligned} E(X) &\leq E(Y) + r + E(Z) \\ &\leq T_r + r + P_k \cdot T_D \end{aligned}$$

The worst running time of the protocol is

$$r \cdot k + T_D$$

Note that by the Markov inequality it is always true that

$$P_k \leq \frac{T_R}{k \cdot r}$$

However in all known randomized protocols a much stronger statement holds; namely, for some constant $c \quad 1$,

$$P_k \leq c^k$$

In particular, let us use a standard deterministic protocol (such as in [MW]) with $T_D = t+1$, and the randomized protocol suggested in [FM] with $r = 15,\ \ T_R = 56 \ \ and \ \ P_k \leq 0.27^{k-1}$. Finally, choosing $k = O(\log\ t)$ gives a constant expected time and worst case time of $t + O(\log\ t)$.

6 The Asynchronous Case

Fischer, Lynch and Peterson [FLP] proved that no deterministic solution exists for the asynchronous BA problem even with one faulty processor. It follows that there is no randomized asynchronous protocol that terminates for all coin-tosses, all inputs, and all adversary behavior. Otherwise choose in advance a sequence of coin-tosses, say - all zeros, and get a deterministic protocol that always terminates contradicting [FLP]. Therefore, in the asynchronous case it is not possible to "benefit from both worlds".

7 Conclusions and Open Problems

In light of the result presented here, even the greatest opposers of randomization must agree to use it for Byzantine Agreement. There is almost nothing to loose. If worse comes to worst, we end up with essentially the same complexity as in the deterministic case, whereas if we are even slightly "lucky" we gain a lot !

The choice of k introduces a trade-off between worst case and average case: the worst case increases linearly with k, while the average case decreases exponentially with k. Can a better trade-off be found ? In particular, can one obtain a smaller (than $\log t$) addition to T_D for the worst case, while keeping the expected time a constant?

8 References

[Be] M. Ben-Or, "Advantages of Free Choice: Completely Asynchronous Agreement Protocols", 1983 *PODC,* pp. 27-30.

[Br] G. Bracha, "An $O(\log n)$ Expected Rounds Randomized Byzantine Generals Protocol", *JACM 34(4),* pp. 910-920

[CC] B. Chor and B. Coan, "A Simple and Efficient Randomized Byzantine Agreement Algorithm", *IEEE Transactions on Software Engineering,* Vol. SE-11, No. 6 1985.

[CMS] B. Chor, M. Meritt and D.Shmoys, "Simple Constant-Time Consensus Protocols in Realistic Failure Models", 1985 *PODC.*

[DM] C. Dwork and Y. Moses, "Knowledge and Common Knowledge in a Byzantine Environment: The Case of Crash Failures", *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge, Monterey 1986, J.Y. Halpern ed., Morgan Kaufman,* pp. 149-170.

[FM] P. Feldman and S. Micali, "Optimal Algorithms for Byzantine Agreement", 1988 *FOCS,* pp. 148-161.

[FLP] M. Fischer, N. Lynch and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process", *JACM* 32(2), pp. 374-382, 1985.

[FL] M. Fischer and N. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency", *Information Processing Letters,* 14(4), pp. 183-186, 1982.

[MT] Y. Moses and M.R. Tuttle, "Programming Simultaneous Actions Using Common Knowledge: Preliminary Version", 1986 *FOCS,* pp. 208-221

[MW] Y. Moses and O. Waatz, "Coordinated Traversal: $(t+1)$-Round Byzantine Agreement in Polynomial Time", 1988 *FOCS.*

[PSL] M. Pease, R.Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults", *JACM* 27(2), 1980.

[R] M. Rabin, "Randomized Byzantine Generals", 1983 FOCS, pp. 403-409.

[TC] R. Turpin and B. Coan, "Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement", *Information Processing Letters,* Vol. 18, pp. 73-76, Feb. 1984.