The Complexity of Early Deciding Set Agreement: How can Topology help?

Rachid Guerraoui

Distributed Programming Laboratory
EPFL
Switzerland

Bastian Pochon

Distributed Programming Laboratory
EPFL
Switzerland

Abstract

The aim of this paper is to pose a challenge to the experts of (algebraic) topology techniques. We present an early deciding algorithm that solves the set agreement problem, i.e., the problem which triggered research on applying topology techniques to distributed computing. We conjecture the algorithm to be optimal, and we discuss the need and challenges of applying topology techniques to prove the lower bound.

Key words: Distributed algorithms, algebraic topology, set agreement.

1 Introduction

Results about the set agreement problem are intriguing, in the sense that they present an intrinsic trade-off between the number of processes in a system, the degree of coordination that these processes can reach, and the number of failures that can be tolerated [3]. Set agreement is a generalization of the widely studied consensus problem [4], in which each process is supposed to propose a value, and eventually decide on some value that was initially proposed, such that every correct eventually decides (just like in consensus). In contrast with consensus however, processes may not decide on more than k distinct values. Hence set agreement is also referred to as k-set agreement.

K-set agreement was introduced in [2]. The paper also introduced k-set

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

agreement algorithms in the asynchronous model [1] when less than k processes may crash. In [7], techniques borrowed from algebraic topology were first used to prove the impossibility of k-set agreement in an asynchronous model where k processes may crash. In [3,6], tight lower bounds were derived for set agreement in the synchronous model prone to process crash. The framework presented in [6] uses the tools from algebraic topology introduced in [7] and allows for proving lower bounds in both the asynchronous and the synchronous models.

Early deciding algorithms are those the efficiency of which depends on the effective number of failures in a given run, rather than on the (total) number of failures that can be tolerated. The effective number of failures is traditionally denoted by f, whereas the total number of failures that are tolerated is denoted by t. In practice, failures rarely happen, and it makes sense to devise algorithms that decide earlier when fewer failures occur. For uniform consensus, Charron-Bost and Schiper [1] have shown that there is a significant improvement on the efficiency when considering the effective number of failures. More precisely, they propose a uniform consensus algorithm in which every process that decides, decides by round $f + 2$ in any run with f failures. This bound is shown to be tight [1,8].

To the best of our knowledge, no result for set agreement have been presented in the context of early deciding algorithms. In the present paper, we give an early deciding set agreement algorithm. We discuss the need and challenges of applying topology techniques to prove optimality of the algorithm.

The rest of the paper is organized as follows. Section 2 gives our system model. Section 3 presents our early deciding algorithm. Section 4 discusses the optimality of this result.

2 Model

We consider a set of $N = n + 1$ processes $\Pi = \{p_0, \ldots, p_n\}$. Processes communicate by message-passing. We consider that communication channels are reliable. Processes execute in a synchronous, round-based model [9]. A run is a sequence of rounds. Every round is composed of three phases. In the first phase, every process broadcasts a message to all the other processes. In the second phase, every process receives all the messages sent to it during the round. In the third phase, every process may perform a local computation, before starting the next round. Processes may fail by crashing. A process that crashes does not execute any step, and is said to be *faulty*. Processes that do not crash are said to be *correct*. When process p_i crashes in round r, a subset of the messages that p_i sends in round r (possibly the empty set) is received by the end of round r. A message broadcast in round r by a process

[1] In the asynchronous model, there is no bound on process relative speed and message communication delay.

that does not crash in round r is received, at the end of round r, by every process that reaches the end of round r. We consider that there are at most $t < N$ processes that may fail in any run.

3 Algorithm

Figure 1 presents an early deciding k-set agreement algorithm. For $t < N - k$ (or equivalently, $t \leq n - k$), this algorithm achieves the following bounds: (1) for $0 \leq \lfloor f/k \rfloor \leq \lfloor t/k \rfloor - 2$, every process that decides, decides by round $\lfloor f/k \rfloor + 2$, and (2) for $\lfloor f/k \rfloor \geq \lfloor t/k \rfloor - 1$, every process that decides, decides by round $\lfloor f/k \rfloor + 1$. Note that this is a strict generalization of the tight lower bounds on uniform consensus [1,8]. [2]

In the algorithm, every process p_i sends its estimate value est_i in every round. At the end of every round, p_i updates est_i with the minimum estimate value received from any other process. The intuition behind set agreement achieved by the algorithm is as follows. In round r, if p_i observes that $k - 1$ processes, or less, crash in that round, then process p_i knows all but at most $k - 1$ values among the smallest values remaining in the system. Process p_i can thus safely decide on est_i if p_i reaches the end of the next round.

We give an intuition of why the algorithm is faster when $\lfloor f/k \rfloor = \lfloor t/k \rfloor - 1$. Note that in this case, every process that decides, decides by round $\lfloor f/k \rfloor + 1$. At the end of round $\lfloor t/k \rfloor - 1$, the processes have more than k distinct estimate values only if there remain $2k - 1$ processes or less that are still allowed to crash. In round $\lfloor t/k \rfloor - 1$, every process that detects $k - 1$ or less new crashes may safely decide at the end of round $\lfloor t/k \rfloor$. The reason is the following. First, if $k - 1$ or less processes crash in round $\lfloor t/k \rfloor$, then at most $k - 1$ distinct estimate values remain in the system, and it is safe to decide for any process. In contrast, if more than $k - 1$ processes crash in round $\lfloor t/k \rfloor$, then $k - 1$ or less processes may still crash. Denote by x the number of processes that detect less than $k - 1$ process crashes in round $\lfloor t/k \rfloor$. These x processes decide at the end of round $\lfloor t/k \rfloor$. Assume that they immediately crash after deciding. Thus there are at most $k - 1 - x$ processes that may still crash in the last round $\lfloor t/k \rfloor + 1$. At the end of that round $\lfloor t/k \rfloor + 1$, at most $k - x$ values may be decided (if $k - 1 - x$ processes crash). In total, processes decide at most on $x + (k - x)$ distinct values.

In the following proofs, we denote the local copy of a variable var at process p_i by var_i, and the value of var_i at the end of round r by var_i^r. $crashed^r$ denotes the set of processes that crash *before* completing round r, $ests^r$ denotes the set of estimate values of every process at the end of round r. By definition, round 0 ends when the algorithm starts. No process decides by round 0. We first prove three general claims about the algorithm of Figure 1.

[2] For uniform consensus, the tight lower bound is $f + 2$, for $0 \leq f \leq t - 2$, and $f + 1$, for $f \geq t - 1$ [1].

```
At process p_i:
 1: halt := ∅ ; decided := deciding := false
 2: S^r := ∅, 1 ≤ r ≤ ⌊t/k⌋ + 1

 3: procedure propose(v_i)
 4:    est_i := v_i
 5:    for r from 1 to ⌊t/k⌋ + 1 do
 6:       if decided or deciding then send (r, DEC, est_i) to all
 7:       else send (r, EST, est_i) to all
 8:       if deciding then
 9:          decide(est_i) ; return
10:       else if decided then
11:          return
12:       else if received any (r, DEC, est_j) then
13:          est_i := est_j ; deciding := true
14:       else
15:          S^r := {(est_j, j) | (r, EST, est_j) is received in round r from p_j}
16:          halt := Π\ ∪_{(est_j,j)∈S^r} {j}
17:          est_i := min{est_j|(est_j, j) ∈ S^r}
18:          if r = ⌊t/k⌋ and |S^r| ≥ N − k⌊t/k⌋ + 1 then
19:             decided := true ; decide(est_i)
20:          else if |halt| < rk then
21:             deciding := true
22:    decide(est_i)
23:    return
```

Fig. 1. An early deciding k-set agreement algorithm (code for process p_i)

Claim 3.1 *$ests^r \subseteq ests^{r-1}$.*

Proof. The proof of the claim is straightforward: for any process p_i, $est_i^r \in ests^{r-1}$. □

Claim 3.2 *If at the end of round $0 \leq r \leq \lfloor t/k \rfloor$ no process has decided, and at most l processes crash in round $r+1$, then $|ests^{r+1}| \leq l+1$.*

Proof. Consider that the conditions of the claim hold and assume by contradiction that $|ests^{r+1}| \geq l+2$. By assumption, there are $l+2$ processes with distinct estimate values at the end of round $r+1$. Denote by $q_0, \ldots, q_{l+1}$ these processes, such that $est_{q_i}^{r+1} \leq est_{q_{i+1}}^{r+1}$, for $0 \leq i \leq l+1$. Processes $q_0, \ldots, q_l$ do not send $est_{q_0}^{r+1}, \ldots, est_{q_l}^{r+1}$ in round $r+1$; otherwise, q_{l+1} receives one of the smallest $l+1$ estimate values in round $r+1$. Thus there are $l+1$ processes which send values corresponding to $est_{q_0}^{r+1}, \ldots, est_{q_l}^{r+1}$ in round $r+1$ and which crash in round $r+1$; otherwise, q_{l+1} receives one of the smallest $l+1$ estimate value in round $r+1$. This contradicts our assumption that at most l processes crash in round $r+1$. □

Claim 3.3 *If, at the end of round* $1 \leq r \leq \lfloor t/k \rfloor$*, no process has decided, and* $|ests^r| \geq k+1$*, then* $|crashed^r| \geq rk$*.*

Proof. We prove the claim by induction. For the base case $r = 1$, assume that the conditions of the claim hold. That is, at the end of round 1, there exist $k+1$ distinct processes $q_0, \ldots, q_k$ with distinct estimate values. By Claim 3.2, $|crashed^1| \geq k$. Assume the claim for round $r-1$, and assume the conditions of the claim hold at round r. We prove the claim for round r. By assumption, there are $k+1$ processes $q_0, \ldots, q_k$ at the end of round r with $k+1$ distinct estimates. By Claim 3.1, $k+1$ processes necessarily reach the end of round $r-1$ with $k+1$ distinct estimates. Thus Claim 3.3 holds at round $r-1$ (induction hypothesis), and thus, $|crashed^{r-1}| \geq (r-1)k$. By Claim 3.2, at least k processes crash in round r. Thus $|crashed^r| \geq k + |crashed^{r-1}| \geq rk$.□

The next proposition asserts the correctness of the algorithm.

Proposition 3.4 *The algorithm in Fig. 1 solves* k*-set agreement.*

Proof. Validity and Termination are obvious. To prove k-ket agreement, we consider the lowest round r in which some process decides. Let p_i be one of the processes that decides in round r. We consider three mutually exclusive cases: (1) p_i decides in round $2 \leq r \leq \lfloor t/k \rfloor - 1$, (2) p_i decides in round $r = \lfloor t/k \rfloor$, and (3) p_i decides in round $r = \lfloor t/k \rfloor + 1$. (In the algorithm, no process decides before round 2.)

Case 1. p_i necessarily decides at line 9, and thus executes line 21 in round $r-1$, where *deciding* is set to *true*. (Because no process decides before p_i, p_i may not receive any DEC message before deciding; and because $r \leq \lfloor t/k \rfloor - 1$, p_i may not decide at line 19.) In round $r-1$, p_i executes line 21 only if p_i evaluates $|crashed^{r-1}| < rk$ at line 20. Thus, from Claim 3.3, there are at most k distinct estimates at the end of round $r-1$, which ensures agreement.

Case 2. There are two cases to consider: (1) p_i decides at line 9, after executing line 21 at the end of round $r-1$, or (2) p_i decides at line 19. (Because no process decides before p_i, p_i may not receive any DEC message before deciding.) In case (1), p_i executes line 21 in round $r-1$ only if p_i evaluates $|crashed^{r-1}| < rk$ at line 20. Thus, from Claim 3.3, there are at most k distinct estimates at the end of round $r-1$, which ensures agreement. In case (2), we consider $ests^{r-1}$. If $|ests^{r-1}| \leq k$, agreement is ensured thereafter. Thus consider that $|ests^{r-1}| \geq k+1$. By Claim 3.3, there exist $k+1$ distinct processes with different estimates at the end of round $r-1$ only if $|crashed^{r-1}| \geq k(r-1) = k(\lfloor t/k \rfloor - 1) \geq t - 2k + 1$, or, equivalently, only if at most $2k-1$ processes may crash in the two subsequent rounds (rounds $\lfloor t/k \rfloor$ and $\lfloor t/k \rfloor + 1$). In round $\lfloor t/k \rfloor$, p_i decides at line 19 only if p_i receives at least $n - k\lfloor t/k \rfloor + 1$ messages. Thus, by Claim 3.2, the processes that decide at the end of round $\lfloor t/k \rfloor$, including p_i, decide on at most k distinct values. Denote by x the number of processes that effectively crash in round $\lfloor t/k \rfloor$, and by y the number of processes that decide at the end of round $\lfloor t/k \rfloor$. We distinguish two cases: (a)

$x \leq k-1$, and (b) $x \geq k$. In case (a), by Claim 3.2, $k-1$ values or less remain in the system at the end of round $\lfloor t/k \rfloor$; agreement is then ensured. In case (b), at most $2k-1-x \leq k-1$ processes may crash among the processes that decide at the end of round $\lfloor t/k \rfloor$ and the processes that take part to round $\lfloor t/k \rfloor + 1$. We claim that the total number of distinct decision values is at most k. Indeed, denote by y_{crash} the number of processes that decide at the end of round $\lfloor t/k \rfloor$ and then immediately crash. In round $\lfloor t/k \rfloor + 1$, at most $k - 1 - y_{crash}$ may crash. By Claim 3.2 processes that decide at the end of round $\lfloor t/k \rfloor + 1$ may decide on at most $k - y_{crash}$ distinct estimate values. Hence the maximum number of decided values is $(k - y_{crash}) + y_{crash} = k$.

Case 3. By contradiction, consider that, at the end of round $\lfloor t/k \rfloor + 1$, there exist $k+1$ distinct processes $q_0, \ldots, q_k$ with different estimates, and which decide on their estimates. By Claim 3.1, there exist $k+1$ processes with distinct estimates at the end of round $r-1$. By Claim 3.3 and because $r = \lfloor t/k \rfloor + 1$, $|crashed^{r-1}| > k(r-1) = k\lfloor t/k \rfloor > t - k$. By Claim 3.2, there exist k processes that crash in round $\lfloor t/k \rfloor + 1$. Thus $|crashed^{r}| \geq k + |crashed^{r-1}| = k + k\lfloor t/k \rfloor > t$. A contradiction. □

The next proposition asserts the efficiency of the algorithm.

Proposition 3.5 *In any run with $0 \leq f \leq t$ failures, any process that decides, decides*

(i) *by round $\lfloor f/k \rfloor + 2$, if $0 \leq \lfloor f/k \rfloor \leq \lfloor t/k \rfloor - 2$, and*

(ii) *by round $\lfloor f/k \rfloor + 1$, if $\lfloor f/k \rfloor \geq \lfloor t/k \rfloor - 1$.*

Proof. We proceed by separating both cases.

Case i. Assume a run with f failures, such that $\lfloor f/k \rfloor \leq \lfloor t/k \rfloor - 2$. By contradiction, assume that there exists a process p_i for which $|halt_i^r| \geq rk$, for $r = \lfloor f/k \rfloor + 1$. (If $|halt_i^r| < rk$, then p_i decides at line 9 in the next round.) Process p_i does not decide in round r; in particular, p_i does not receive any DEC message in round r. We have $|halt_i^r| \geq rk = (\lfloor f/k \rfloor + 1)k = \lfloor f/k \rfloor k + k > f$. A contradiction.

Case ii. Assume a run with f failures, such that $\lfloor f/k \rfloor \geq \lfloor t/k \rfloor - 1$. First assume that $\lfloor f/k \rfloor = \lfloor t/k \rfloor - 1$, and assume by contradiction that there exists a process p_i that does not decide by round $r = \lfloor f/k \rfloor + 1$. Thus p_i does not receive any DEC message in round r. Assume by contradiction that p_i does not decide at line 19. Thus $|S^r| < N - k\lfloor t/k \rfloor + 1$, and $f > k\lfloor t/k \rfloor - 1$. This implies in turn that $\lfloor f/k \rfloor > \lfloor t/k \rfloor - 1$. A contradiction. When $\lfloor f/k \rfloor = \lfloor t/k \rfloor$, then any process that decides, decides by round $\lfloor f/k \rfloor + 1 = \lfloor t/k \rfloor + 1$. □

4 Towards the Optimality Proof

For the case $k = 1$, the optimality of our algorithm falls back to the results of uniform consensus, for which the lower bound of $f+2$, for $0 \leq f \leq t-2$, and $f+1$, for $f \geq t-1$, was proven to be tight in [1,8].

In [5], and for $k > 1$, we propose a proof of optimality based on the reduction of our problem to set agreement in the asynchronous model, which was proven to be impossible [7]. More precisely, we prove that, for any value of k and f, there exists no algorithm such that (i) a process that sees f failures, decides at the end of $\lfloor f/k \rfloor + 1$, and (ii) in runs in which eventually no more than $k-1$ processes fail in each round, every correct process eventually decide. Whereas it is impossible to design a uniform consensus algorithm, in which any process decides after round 1, even in a failure-free execution, it is easy to see that it is possible to design a k-set agreement algorithm, in which any process that sees no failure (or, more generally, that sees $k-1$ failures or less) at the end of round 1, decides, and eventually every correct process decides (not necessarily at the end of round $\lfloor t/k \rfloor + 1$).

Our proof in [5] does not directly rely on algebraic topology. Roughly speaking, we reduce the problem of early deciding set agreement in a synchronous, message-passing model, to the problem of set agreement in an asynchronous, shared-memory model, and show how this implies a contradiction. We introduce a simulation algorithm (detailed in [5]), which enables processes in a synchronous, message-passing model, to simulate, with the help of a (supposely existing) set agreement algorithm satisfying (i) and (ii) above, an execution of a wait-free [3] set agreement algorithm in an asynchronous, shared-memory model. In [7], wait-free set agreement was proven to be impossible in an asynchronous, shared-memory model. This leads to the desired contradiction.

The remaining part of the optimality is still left open. It consists in proving that, for any value of k, and any value of $\lfloor f/k \rfloor$, no algorithm can decide in $\lfloor f/k \rfloor + 1$ rounds. In this case, we envisage a proof along the lines of [7,6], based on notions of algebraic topology. We discuss why the techniques presented in [7,6] do not however directly apply, and we propose a possible line of research to address this open question.

The principle behind the proofs in [7,6] is (1) to associate a so-called *protocol complex* to the set of all executions of the processes of a full-information protocol in a given model, and (2) to observe that such a protocol complex presents a topological obstruction that prevents it to be mapped onto the output complex of k-set agreement. [4] The abstraction that is used is $(k-1)$-connectivity. Indeed, Theorem 6 in [6] relates the $(k-1)$-connectivity of a protocol complex for k-set agreement in any model, with the impossibility of solving k-set agreement in that model.

Connectivity leads to impossibility because we assume that the processes all need to decide at the end of the same round. Indeed, one can apply Sperner's lemma to show that there exists at least one execution where more than k values are decided, when the protocol complex is $(k-1)$-connected [7,6].

[3] In a wait-free implementation of an algorithm, all processes but one may fail.

[4] The output complex represents the set of all possible final states of the processes, according to the specification of k-set agreement.

On the other hand, in the algorithm presented in this paper, the processes may actually decide faster than $\lfloor t/k \rfloor + 1$, the tight lower bound for k-set agreement [3,6]. [5] Why is that possible? Is there any contradiction?

In fact, there is no contradiction. Processes may actually decide faster than the lower bound of $\lfloor t/k \rfloor + 1$, because, in the early deciding case, the processes are not forced to necessarily decide *all* at the end of the *same* round. In other word, even so the protocol complex is still $(k-1)$-connected after, say, round $r < \lfloor t/k \rfloor + 1$, *some* processes may already decide, provided that these processes span a subcomplex within the full protocol complex that is, at least, not $(k-1)$-connected.

References

[1] B. Charron-Bost and A. Schiper. Uniform consensus harder than consensus. Technical Report DSC/2000/028, École Polytechnique Fédérale de Lausanne, Switzerland, May 2000.

[2] S. Chaudhuri. More choices allow more faults: set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, July 1993.

[3] S. Chaudhuri, M. Herlihy, N. A. Lynch, and M. R. Tuttle. Tight bounds for k-set agreement. *Journal of the ACM*, 47(5):912–943, 2000.

[4] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[5] E. Gafni, R. Guerraoui, and B. Pochon. From a static impossibility to an adaptive lower bound: the complexity of early deciding set agreement. In *Proceedings of the* 37^{th} *ACM Symposium on Theory of Computing (to appear)*, May 2005.

[6] M. Herlihy, S. Rajsbaum, and M. Tuttle. Unifying synchronous and asynchronous message-passing models. In *Proceedings of the* 17^{th} *ACM Symposium on Principles of Distributed Computing*, pages 133–142, 1998.

[7] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858–923, 1999.

[8] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults – a tutorial. Technical report, MIT Technical Report MIT-LCS-TR-821, 2001. (Preliminary version in SIGACT News, Distributed Computing Column, 32(2):45–63, 2001).

[9] N. A. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.

[5] For example, in the failure-free run, all processes decide by the end of round 2, for any t and any k.