Narrowing Power vs. Efficiency in Synchronous Set Agreement

Achour Mostefaoui, Michel Raynal, and Corentin Travers

IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
{achour,raynal,ctravers}@irisa.fr

Abstract. The k-set agreement problem is a generalization of the uniform consensus problem: each process proposes a value, and each non-faulty process has to decide a value such that a decided value is a proposed value, and at most k different values are decided. It has been shown that any algorithm that solves the k-set agreement problem in synchronous systems that can suffer up to t crash failures requires $\lfloor \frac{t}{k} \rfloor + 1$ rounds in the worst case. It has also been shown that it is possible to design early deciding algorithms where no process decides and halts after $\min\left(\lfloor \frac{f}{k} \rfloor + 2, \lfloor \frac{t}{k} \rfloor + 1\right)$ rounds, where f is the number of actual crashes in a run ($0 \leq f \leq t$).

This paper explores a new direction to solve the k-set agreement problem in a synchronous system. It considers that the system is enriched with base objects (denoted $[m, \ell]$_SA objects) that allow solving the ℓ-set agreement problem in a set of m processes ($m < n$). The paper has several contributions. It first proposes a synchronous k-set agreement algorithm that benefits from such underlying base objects. This algorithm requires $O(\frac{t\ell}{mk})$ rounds, more precisely, $R_t = \lfloor \frac{t}{\Delta} \rfloor + 1$ rounds, where $\Delta = m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell)$. The paper then shows that this bound, that involves all the parameters that characterize both the problem (k) and its environment (t, m and ℓ), is a lower bound. The proof of this lower bound sheds additional light on the deep connection between synchronous efficiency and asynchronous computability. Finally, the paper extends its investigation to the early deciding case. It presents a k-set agreement algorithm that directs the processes to decide and stop by round $R_f = \min\left(\lfloor \frac{f}{\Delta} \rfloor + 2, \lfloor \frac{t}{\Delta} \rfloor + 1\right)$. These bounds generalize the bounds previously established for solving the k-set problem in pure synchronous systems.

1 Introduction

Context of the work. The k-set agreement problem generalizes the uniform consensus problem (that corresponds to the case $k = 1$). That problem has been introduced by S. Chaudhuri to investigate how the number of choices (k) allowed to the processes is related to the maximum number (t) of processes that can crash during a run [4]. The problem can be defined as follows. Each of the n processors (processes) defining the system starts with a value (called a "proposed" value). Each process that does not crash has to decide a value (termination), in such a way that a decided value

S. Rao et al. (Eds.): ICDCN 2008, LNCS 4904, pp. 99–111, 2008.

is a proposed value (validity), and no more than k different values are decided (agreement)[1].

When we consider asynchronous systems, the problem can trivially be solved when $k > t$. Differently, it has been shown that there is no solution in these systems as soon as $k \leq t$ [3,14,23]. (The asynchronous consensus impossibility, case $k = 1$, was demonstrated before using a different technique). Several approaches have been proposed to circumvent the impossibility to solve the k-set agreement problem in asynchronous systems (e.g., probabilistic protocols [20], unreliable failure detectors with limited scope accuracy [12,19], or conditions associated with input vectors [17]).

The situation is different in synchronous systems where the k-set agreement problem can always be solved, whatever the respective values of t and k. This has an inherent cost, namely, the smallest number of rounds (time complexity measured in communication steps) that have to be executed in the worst case scenario is lower bounded by $\lfloor \frac{t}{k} \rfloor + 1$ [5]. (That bound generalizes the $t+1$ lower bound associated with the consensus problem [1,7].)

Although failures do occur, they are rare in practice. For the uniform consensus problem ($k = 1$), this observation has motivated the design of early deciding synchronous protocols [6,15], i.e., protocols that can cope with up to t process crashes, but decide in less than $t + 1$ rounds in favorable circumstances (i.e., when there are few failures). More precisely, these protocols allow the processes to decide in $\min(f + 2, t + 1)$ rounds, where f is the number of processes that crash during a run, $0 \leq f \leq t$, which has been shown to be optimal (the worst scenario being when there is exactly one crash per round).

In a very interesting way, it has also been shown that the early deciding lower bound for the k-set agreement problem is $\min(\lfloor \frac{f}{k} \rfloor + 2, \lfloor \frac{t}{k} \rfloor + 1)$ [10]. This lower bound, not only generalizes the corresponding uniform consensus lower bound, but also shows an "inescapable tradeoff" among the number t of faults tolerated, the number f of actual faults, the degree k of coordination we want to achieve, and the best running time achievable. It is important to notice that, when compared to consensus, k-set agreement divides the running time by k (e.g., allowing two values to be decided halves the running time).

Related work. To our knowledge, two approaches have been proposed and investigated to circumvent the $\min(\lfloor \frac{f}{k} \rfloor + 2, \lfloor \frac{t}{k} \rfloor + 1)$ lower bound associated with the synchronous k-set agreement problem.

The first is the *fast failure detector* approach that has been proposed and developed in [2] to expedite decision in synchronous consensus. That approach assumes a special hardware that allows a process to detect the crash of any process at most d time units after the crash occurred, where $d < D$, D being the maximum message delay provided by the synchronous system. Both d and D are a priori known by the processes. A fast failure detector-based consensus algorithm that terminates in $D + fd$ is proposed in [2], where it is also shown that $D + fd$ is a lower bound for any algorithm based on a

[1] This paper considers the crash failure model. The reader interested by the k-set agreement problem in more severe send/receive/general omission failure models can consult the introductory survey [22].

fast failure detector[2]. To our knowledge, this approach has been considered only for the consensus problem.

A second approach that has been proposed to circumvent the $\min(f+2, t+1)$ lower bound is the use of conditions [18]. That approach considers that the values proposed by the processes define an input vector with one entry per process. Basically, a *condition* C_t^d (t and d are two parameters that allow defining instances of the condition) is a set of input vectors I such that $\forall I \in C_t^d$, there is a value that appears in I more than $t-d$ times. A deterministic way to define which value has to appear enough times in a vector I (e.g., the maximal value of the vector [16]) allows defining a hierarchy of conditions such that $C_t^0 \subset \cdots \subset C_t^x \subset \cdots \subset C_t^t$ (where C_t^t is the condition including all the input vectors).

[18] presents two main results. Let I be the input vector of the considered run, and C_t^d be a condition. The first result is a synchronous consensus algorithm that allows the processes to decide in (1) one round when $I \in C_t^d$ and $f = 0$, (2) two rounds when $I \in C_t^d$ and $f \leq t-d$, (3) $\min(d+1, f+2, t+1)$ rounds when $I \in C_t^d$ and $f > t-d$, and (4) $\min(f+2, t+1)$ when $I \notin C_t^d$. The second result is a proof showing that $\min(d+1, f+2, t+1)$ rounds are necessary in the worst case when $I \in C_t^d$ (and $I \notin C_t^{d-1}$).

Problem addressed in the paper. The paper is on the efficiency (measured as the number of rounds required to decide) of synchronous set agreement algorithms. As it has just been shown, fast failure detectors and conditions are two ways to circumvent the synchronous lower bound. The paper investigates a third approach. That approach is based on base objects that allow narrowing the set of proposed values. Their aim is to play a part similar to fast failure detectors or conditions, i.e., allow expediting consensus.

Let us consider as a simple example a test&set object. This object has consensus number 2 [11], which means that it allows solving consensus in an asynchronous system made up of two processes (where one of them can crash), but not in a system made up of $n > 2$ processes (where up to $n-1$ can crash)[3]. Is it possible to use such base objects to speed up synchronous set agreement in a system made up of n processes where up to t may crash? More generally, let $[m, \ell]$_SA denote an object that allows solving ℓ-set agreement in a synchronous system of m processes. As fast failure detectors or conditions, these objects are assumed given for free. So, the previous question becomes:

- Is it possible to benefit from $[m, \ell]$_SA objects to build a t-resilient synchronous $[n, k]$_SA object (i.e., a k-set agreement object that has to cope with up to t process crashes)?
- If such a construction is possible, is its cost smaller than $\lfloor \frac{t}{k} \rfloor + 1$, or smaller than $\min(\lfloor \frac{f}{k} \rfloor + 2, \lfloor \frac{t}{k} \rfloor + 1)$ if we are interested in an early deciding $[n, k]$_SA object?

If m, ℓ, n and k are such that there is an integer a with $n \leq am$ and $a\ell \leq k$, it is possible to solve the k-set agreement problem without exchanging any value (i.e., in 0

[2] Without a fast failure detector, the cost would be $D \times \min(f+2, t+1)$.

[3] The consensus number of a concurrent object type is the maximum number of processes that can solve consensus (despite any number of process crashes) using only atomic registers and objects of that type. The consensus number of test&set objects, queues, and stacks is 2 [11].

round!) whatever the value of t. This is trivially obtained by partitioning the n processes into a subsets of at most m processes, and using in each subset a $[m, \ell]$_SA object in order that each process be provided with a decided value. So, the interesting cases are when the values m, ℓ, n and k do not allow a trivial partitioning such as the previous one.

Another way to present the previous question is the following: how much crashes can we tolerate when we want to build a synchronous $[10, 3]$_SA object from $[2, 1]$_SA objects, if one wants to decide in at most one round? In at most two rounds? In at most three rounds?

From a more practical point of view, we can see the system as made up of clusters of m processes, such that an operation involving only processes of a given cluster can be performed very efficiently, i.e., in a time that is smaller than the maximal message transfer delay involving processes belonging to different clusters. That is the sense in which the sentence "the $[m, \ell]$_SA objects are given for free" has to be understood.

Results. The paper presents the following results.

- It first presents a synchronous message-passing algorithm that builds a $[n, k]$_SA object from $[m, \ell]$_SA objects. This algorithm works for any values of n, k, m, and ℓ (assuming, of course, $n > k$ and $m > \ell$).
- The paper then shows that the number of rounds (R_t) of the previous algorithm varies as $O(\frac{t\ell}{mk})$. This means that R_t (1) decreases when the coordination degree k increases (i.e., when less synchronization is required), or when the number of processes m involved in each underlying object increases, and (2) increases when the underlying object is less and less powerful (i.e., when ℓ increases) or when the number of process crashes that the algorithm has to tolerate increases. More precisely, we have:

$$R_t = \lfloor \frac{t}{m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell)} \rfloor + 1.$$

  When we consider the previous example of building, in a synchronous system, a $[10, 3]$_SA object from $[2, 1]$_SA objects, we can conclude that $R_t = 1$ requires $t < 6$, while $R_t = 2$ allows $t = 9$. Moreover, as there are only $n = 10$ processes, there is no value of t that can entail an execution in which $R_t = 3$ are required (for it to occur, we should have $12 \leq t < 18$ and $n > t$).

  To have a better view of R_t, it is interesting to look at special cases.

  - Case 1. Build a consensus object in a synchronous system from $[1, 1]$_SA base objects or $[m, m]$_SA objects (i.e., from base objects that have no power). It is easy to see that $R_t = t+1$ (that is the well-known lower bound for synchronous t-resilient consensus).
  - Case 2. Build a $[n, k]$_SA object in a synchronous system from $[1, 1]$_SA base objects or $[m, m]$_SA objects (base objects without power). It is easy to see that $R_t = \lfloor \frac{t}{k} \rfloor + 1$, (that is the lower bound for synchronous t-resilient k-set agreement).
  - Case 3. Build a synchronous consensus from $[m, 1]$_SA base objects (i.e., consensus objects). In that case $R_t = \lfloor \frac{t}{m} \rfloor + 1$.

- Case 4. Build a synchronous $[n, \ell]$_SA object from $[m, \ell]$_SA base objects. In that case, $R_t = \lfloor \frac{t}{m} \rfloor + 1$.
- Case 5. Build a synchronous $[n, k]$_SA object from $[m, 1]$_SA base objects (i.e., consensus objects). We then have $R_t = \lfloor \frac{t}{mk} \rfloor + 1$.

These particular instances show clearly how the coordination degree and the size of the base objects (measured by the value m) affect the maximal number of rounds executed by the algorithm and consequently allow expediting the decision.

– The paper then shows that the value R_t is optimal when, one wants to build, in a synchronous system, an $[n, k]$_SA object from $[m, \ell]$_SA base objects. This optimality result generalizes previous lower bounds proved for special cases such as consensus [1,7,15], and set agreement [5].

  The optimality proof relies on two theorems, one from Gafni [9], the other from Herlihy and Rajsbaum [13]. Gafni's theorem establishes a deep connection between solvability in asynchronous system and lower bounds (efficiency) in synchronous systems. Herlihy and Rajsbaum's theorem is on the impossibility to solve some set agreement problems in asynchronous systems.

– Finally, the paper extends the algorithm to the early decision case. More specifically, the maximal number of rounds of the early deciding version of the algorithm is the following:

$$R_f = \min\big(\lfloor \frac{f}{\Delta} \rfloor + 2, \lfloor \frac{t}{\Delta} \rfloor + 1\big) \quad \text{where} \quad \Delta = m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell).$$

  It is easy to see that this early decision bound generalizes the lower bounds that are known for the special consensus and set agreement cases.

This paper is an endeavor to capture the essence of the synchronous set agreement and provide the reader with a better understanding of it. To that end, it considers design simplicity as a first-class citizen when both designing algorithms and proving lower bound results[4].

As already noticed, the lower bound proof relies on previous theorems. We do think that Gafni's theorem [9] (that states that an asynchronous system with at most t' crashes can implement the first $\lfloor \frac{t}{t'} \rfloor$ rounds of a synchronous system with up to t failures) is a fundamental theorem of fault-tolerant distributed computing. The lower bound proof of this paper paper shows an application of this powerful theorem.

Roadmap. The paper is made up of 5 sections. Section 2 introduces the system model and definitions. Section 3 presents the algorithm that builds an $[n, k]$_SA object from $[m, \ell]$_SA objects in R_t synchronous rounds. Section 4 proves that R_t is a lower bound on the number of rounds for any synchronous algorithm that builds an $[n, k]$_SA object from $[m, \ell]$_SA objects. Section 5 considers the early decision case.

2 Computation Model and the Set Agreement Problem

The k-set agreement problem. The problem has been informally stated in the Introduction: every process p_i *proposes* a value v_i and each correct process has to *decide* on

[4] The paper strives to modestly follow Einstein's advice "Make it as simple as possible, but no more".

a value in relation to the set of proposed values. More precisely, the *k-set agreement* problem [4] is defined by the following three properties (as we can see 1-set agreement is the uniform consensus problem):

- Termination: Every correct process eventually decides.
- Validity: If a process decides v, then v was proposed by some process.
- Agreement: No more than k different values are decided.

Process model. The system model consists of a finite set of n processes, namely, $\Pi = \{p_1, \ldots, p_n\}$. A process is a sequence of steps (execution of a base atomic operation). A process is *faulty* during an execution if it stops executing steps (after it has crashed a process executes no step). As already indicated, t is an upper bound on the number of faulty processes, while f denotes the number of processes that crash during a particular run, $0 \leq f \leq t < n$. (Without loss of generality we consider that the execution of a step by a process takes no time.)

In the following, we implicitly assume $k \leq t$. This is because k-set agreement can trivially be solved in synchronous or asynchronous systems when $t < k$ [4].

Communication/coordination model. The processes communicate by sending and receiving messages through channels. Every pair of processes p_i and p_j is connected by a channel. The sending of a message and the reception of a message are atomic operations. The underlying communication system is assumed to be failure-free: there is no creation, alteration, loss or duplication of message.

In addition to messages, the processes can coordinate by accessing $[m, \ell]$_SA objects. Such an object is a one-shot object that can be accessed by at most m processes. Its power is to solve the ℓ-set agreement problem among m processes. Let us observe that, for $1 \leq m \leq n$, an $[m, m]$_SA object is a trivial object that has no coordination power.

Round-based synchrony. The system is *synchronous*. This means that each of its runs consists of a sequence of *rounds*. Those are identified by the successive integers $1, 2,$ etc. For the processes, the current round number appears as a global variable r that they can read, and whose progress is given for free: it is managed by an external entity. A round is made up of two main consecutive phases:

- A send phase in which each process sends zero or one message to each other processes. If a process crashes during the send phase of a round, an arbitrary subset of the processes to which it sent messages will receive these messages.
- A receive phase in which each process receives messages. The fundamental property of the synchronous model lies in the fact that a message sent by a process p_i to a process p_j at round r, is received by p_j at the very same round r.

Before or after a phase, a process can execute local computations (e.g., process the messages it received during the current round). It can also invokes an underlying $[m, \ell]$_SA base object.

3 A Synchronous $[n, k]$_SA Algorithm

This section presents a simple algorithm that, when at most t processes may crash, builds an $[n, k]$_SA object if the system provides the n processes with round-based synchrony and $[m, \ell]$_SA base objects.

Notation. In all the rest of the paper we are using the following notations:

- $k = \alpha\ell + \beta$ with $\alpha = \lfloor \frac{k}{\ell} \rfloor$ and $\beta = k \bmod \ell$.
- $\Delta = \alpha\, m + \beta$ and $R_t = \lfloor \frac{t}{\Delta} \rfloor + 1 = \lfloor \frac{t}{m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell)} \rfloor + 1$.

3.1 The Algorithm

The algorithm is pretty simple. It is described in Figure 1. A process p_i invokes the operation $\mathsf{propose}_k(v_i)$ where v_i is the value it proposes. That value is initially stored in the local variable est_i (line 01), that afterwards will contain the current estimate of p_i's decision value (line 10). The process terminates when it executes the $\mathsf{return}(est_i)$ statement.

Each process executes R_t rounds (line 02). During any round r, only Δ processes are allowed to send their current estimates. These processes are called the *senders* of round r. When $r = 1$, they are the processes $p_1, \ldots, p_\Delta$, during the second round the processes $p_{\Delta+1}, \ldots, p_{2\Delta}$, and so on (lines 04-05).

The Δ senders of a round r are partitioned into $\lceil \frac{\Delta}{m} \rceil$ subsets of m processes (the last subset containing possibly less than m processes), and each subset uses an $[m, \ell]$_SA object to narrow the set of its current estimates (lines 06-07). After this "narrowing", each sender process sends its new current estimate to all the processes. A process p_i accesses an $[m, \ell]$_SA object by invoking the operation $\mathsf{propose}(est_i)$. The $\lceil \frac{\Delta}{m} \rceil$ $[m, \ell]$_SA objects used during a round r are in the array $SA[r, 0..\lceil \frac{\Delta}{m} \rceil - 1]$ [5]. Finally, when during a round, a process p_i receives estimates, it updates est_i accordingly (line 10).

It is important to see that, if during a round, at least one sender process does not crash, at most $k = \alpha\ell + \beta$ estimates are sent during that round, which means that k-set agreement is guaranteed as soon as there is a round during which an active process does not crash.

3.2 Proof of the Algorithm

Lemma 1. *Let $nc[r]$ be the number of processes that crash during the round r. There is a round r such that $r \leq R_t$ and $nc[r] < \Delta$.*

Proof. Let $t = \alpha'\Delta + \beta'$ with $\alpha' = \lfloor \frac{t}{\Delta} \rfloor$ and $\beta' = t \bmod \Delta$. The proof is by contradiction. let us assume that, $\forall\, r \leq R_t$, we have $nc[r] \geq \Delta$. We then have:

$$\sum_{r=1}^{R_t} nc[r] \geq \Delta \times R_t = \Delta\big(\lfloor \frac{t}{\Delta} \rfloor + 1\big) = \Delta\big(\alpha' + \lfloor \frac{\beta'}{\Delta} \rfloor + 1\big) = \Delta \times \alpha' + \Delta > t.$$

[5] Actually, only $R_t \lfloor \frac{\Delta}{m} \rfloor$ base $[m, \ell]$_SA objects are needed. This follows from the following observation: during each round r, if $\beta \neq 0$, the "last" β sender processes do not need to use such an $[m, \ell]$_SA object because $\beta \leq \ell$. (Let us recall that $0 \leq \beta < \ell$ and Δ is defined as $\alpha\, m + \beta$.)

```
Function propose_k(v_i)
(01) est_i ← v_i;
(02) for r = 1, 2, ..., R_t do % r: round number %
(03) begin round
(04)     first_sender ← (r − 1)Δ + 1; last_sender ← rΔ;
(05)     if first_sender ≤ i ≤ last_sender then % p_i is "sender" at round r %
(06)         let y such that first_sender + ym ≤ i < last_sender + (y + 1)m;
             % y is index of the [m, ℓ]_SA object used by p_i %
(07)         est_i ← SA[r, y].propose(est_i);
(08)         for each j ∈ {1, ..., n} do send (est_i) to p_j end do
(09)     end if;
(10)     est_i ← any est value received if any, unchanged otherwise
(11) end round;
(12) return(est_i)
```

Fig. 1. $[n, k]$_SA object from $[m, \ell]$_SA objects in a synchronous system (code for p_i)

Consequently, there are more than t processes that crash: a contradiction. □

Lemma 2. *At any round r, at most k different estimate values are sent by the processes.*

Proof. Let us recall that $k = \alpha\ \ell + \beta$ (Euclidean division of k by ℓ) and the value Δ is $\alpha\ m + \beta$.

Due to the lines 04-05, at most Δ processes are sender at each round r. These Δ sender processes are partitioned into $\lfloor \frac{\Delta}{m} \rfloor$ sets of exactly m processes plus a set of β processes. As each underlying $[m, \ell]$_SA object used during the round r outputs at most ℓ estimates values from the value it is proposed, it follows that at most $\alpha\ell + \beta$ estimates values can be output by these objects, which proves the lemma. □

Lemma 3. *At most k different values are decided by the processes.*

Proof. At any round the number of senders is at most Δ (lines 04-05). Moreover, due to lemma 1, there is at least one round $r \leq R_t$ during which a correct process is a sender. If follows from Lemma 2, line 08 and line 10, that, at the end of such a round r, the estimates of the processes contain at most k distinct values. □

Theorem 1. *The algorithm described in Figure 1 is a synchronous t-resilient k-set agreement algorithm.*

Proof. The termination property follows directly from the synchrony of the model: a process that does not crash executes R_t rounds. The validity property follows directly from the initialization of the estimate values est_i, the correctness of the underlying $[m, \ell]$_SA objects (line 07), and the fact that the algorithm exchanges only est_i values. Finally, the agreement property is Lemma 3. □

4 Lower Bound on the Number of Rounds

This section proves that the previous algorithm is optimal with respect to the number of rounds. The proof of this lower bound is based on (1) a deep connection relating synchronous efficiency and asynchronous computability in presence of failures [9], and (2) an impossibility result in asynchronous set agreement [13].

4.1 Notation and Previous Results

- $\mathcal{S}_{n,t}[\emptyset]$ denotes the classical round-based synchronous system model made up of n processes, where up to t processes may crash.
- $\mathcal{S}_{n,t}[m,\ell]$ is the $\mathcal{S}_{n,t}[\emptyset]$ system model enriched with $[m,\ell]$_SA objects. This is the model defined in Section 2 (n processes, at most t process crashes, coordination possible through $[m,\ell]$_SA objects).
- $\mathcal{AS}_{n,t}[\emptyset]$ denotes the classical asynchronous system model (n processes, up to processes t may crash, no additional equipment).
- $\mathcal{AS}_{n,t}[m,\ell]$ denotes the asynchronous system model $\mathcal{AS}_{n,t}[\emptyset]$ enriched with $[m,\ell]$_SA objects. (From a computability point of view, $\mathcal{AS}_{n,t}[\emptyset]$ is weaker than $\mathcal{AS}_{n,t}[m,\ell]$.)

The following theorems are central in proving that R_t is a lower bound.

Theorem 2. (Gafni [9]) *Let $n > t \geq k > 0$. It is possible to simulate in $\mathcal{AS}_{n,k}[\emptyset]$ the first $\lfloor \frac{t}{k} \rfloor$ rounds of any algorithm designed for $\mathcal{S}_{n,t}[\emptyset]$ system model.*

The next corollary is a simple extension of Gafni's theorem suited to our needs.

Corollary 1. *Let $n > t \geq k > 0$. It is possible to simulate in $\mathcal{AS}_{n,k}[m,\ell]$ the first $\lfloor \frac{t}{k} \rfloor$ rounds of any algorithm designed for $\mathcal{S}_{n,t}[m,\ell]$ system model.*

Theorem 3. (Herlihy-Rajsbaum [13]) *Let $J_{m,\ell}$ be the function defined as follows: $u \rightarrow \ell \lfloor \frac{u}{m} \rfloor + \min(\ell, u \bmod m) - 1$. There is no algorithm that solves the K-set agreement problem, with $K = J_{m,\ell}(t+1)$, in $\mathcal{AS}_{n,t}[m,\ell]$.*

4.2 The Lower Bound

Theorem 4. *Let $1 \leq \ell \leq m < n$ and $1 \leq k \leq t < n$. Any algorithm that solves the k-set agreement problem in $\mathcal{S}_{n,t}[m,\ell]$ has at least one run in which at least one process does not decide before the round $R_t = \lfloor \frac{t}{m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell)} \rfloor + 1$.*

Proof. The proof is by contradiction. let us assume that there is an algorithm A that solves the k-set agreement problem in at most $R < R_t$ rounds in $\mathcal{S}_{n,t}[m,\ell]$ (this means that any process decides by at most R rounds, or crashes before). We consider two cases.

- $k < \ell$. We have then $R < R_t = \lfloor \frac{t}{k} \rfloor + 1$.

1. As $k < \ell$, the ℓ-set agreement can be solved in in $\mathcal{AS}_{n,k}[\emptyset]$. It follows that, as far as set agreement is concerned, $\mathcal{AS}_{n,k}[\emptyset]$ and $\mathcal{AS}_{n,k}[m,\ell]$ have the same computational power.
2. It follows from the corollary of Gafni's theorem that there is, in $\mathcal{AS}_{n,k}[m,\ell]$, a simulation of the first $\lfloor \frac{t}{k} \rfloor$ rounds of any algorithm designed for the $\mathcal{S}_{n,t}[m,\ell]$ system model. It is consequently possible to simulate in $\mathcal{AS}_{n,k}[m,\ell]$ the $R < R_t = \lfloor \frac{t}{k} \rfloor + 1$ rounds of the algorithm A. It follows that the k-set agreement problem can be solved in in $\mathcal{AS}_{n,k}[m,\ell]$.
3. Combining the two previous items, we obtain an algorithm that solves the k-set agreement problem in $\mathcal{AS}_{n,k}[\emptyset]$. This contradicts the impossibility to solve the k-set agreement problem in $\mathcal{AS}_{n,k}[\emptyset]$ [3,14,23]. This proves the theorem for the case $k < \ell$.

– $k \geq \ell$. Let us recall the definition $\Delta = m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell) = \alpha\, m + \beta$.

1. It follows from the corollary of Gafni's theorem that at least $\lfloor \frac{t}{\Delta} \rfloor$ rounds of any algorithm designed for the $\mathcal{S}_{n,t}[m,\ell]$ system model can be simulated in $\mathcal{AS}_{n,\Delta}[m,\ell]$.

   So, as the algorithm A solves the k-set agreement problem in $\mathcal{S}_{n,t}[m,\ell]$, in at most $R < R_t = \lfloor \frac{t}{\Delta} \rfloor + 1$, combining the simulation with A, we obtain an algorithm that solves the k-set agreement problem in $\mathcal{AS}_{n,\Delta}[m,\ell]$.
2. Considering the argument used in Herlihy-Rajsbaum's theorem we have the following:

$$\begin{aligned} J_{m,\ell}(\Delta+1) &= \ell\, \lfloor \frac{\Delta+1}{m} \rfloor + \min\big(\ell, (\Delta+1) \bmod m\big) - 1, \\ &= \ell\, \lfloor \frac{\alpha\, m + \beta + 1}{m} \rfloor + \min\big(\ell, (\alpha\, m + \beta + 1) \bmod m\big) - 1, \\ &= \ell\, (\alpha + \lfloor \frac{\beta+1}{m} \rfloor) + \min\big(\ell, (\beta+1) \bmod m\big) - 1. \end{aligned}$$

   Let us observe that $\ell \leq m$. Moreover, as $\beta = k \bmod \ell$, we also have $\beta < \ell$. To summarize: $\beta < \ell \leq m$. There are two cases to consider.

   (a) $m = \beta + 1$. Observe that this implies that $\ell = m$ and $\ell - 1 = \beta$.

$$\begin{aligned} J_{m,\ell}(\Delta+1) &= \ell\, (\alpha+1) + \min\big(\ell, m \bmod m\big) - 1, \\ &= \ell\, \alpha + \ell - 1 = \ell\, \alpha + \beta = k. \end{aligned}$$

   (b) $m > \beta + 1$:

$$\begin{aligned} J_{m,\ell}(\Delta+1) &= \ell\, \alpha + \min\big(\ell, (\beta+1) \bmod m\big) - 1, \\ &= \ell\, \alpha + \beta + 1 - 1 = k. \end{aligned}$$

   In both cases, $J_{m,\ell}(\Delta+1) = k$. It follows from Herlihy-Rajsbaum's theorem that there is no algorithm that solves the $J_{m,\ell}(\Delta+1)$-set agreement problem (i.e., the k-set agreement problem) in $\mathcal{AS}_{n,\Delta}[m,\ell]$.
3. The two previous items contradict each other, thereby proving the theorem for the case $k < \ell$. □

```
Function ED_propose_k(v_i)
(01)  est_i ← v_i;
(02)  for r = 1, 2, ..., R_t do % r: round number %
(03)  begin round
(04)      first_sender ← (r − 1)Δ + 1; last_sender ← rΔ;
(05)      if first_sender ≤ i ≤ last_sender then % p_i is "sender" at round r %
(06)          let y such that first_sender + ym ≤ i < last_sender + (y + 1)m;
              % y is index of the [m, ℓ]_SA object used by p_i %
(07)          est_i ← SA[r, y].propose(est_i);
(08)          for each j ∈ {1, ..., n} do send (est_i) to p_j end do
(09)      end if;
(A1)      if (p_i was a sender at round r − 1) then
              for each j ∈ {1, ..., n} do send (COMMIT) to p_j end do end if;
(A2)      if (COMMIT received) then return(est_i) end if;
(10)      est_i ← any est value received if any, unchanged otherwise
(11)  end round;
(12)  return(est_i)
```

Fig. 2. Early-deciding $[n, k]$_SA object from $[m, \ell]$_SA objects in a synchronous system (p_i)

Corollary 2. *When $k < \ell$, the underlying $[m, \ell]$_SA objects are useless.*

Proof. The corollary follows from the fact that $k < \ell \Rightarrow R_t = \lfloor \frac{t}{k} \rfloor + 1$, that is the lower bound when no underlying base object is used. □

This corollary means that no k-set agreement algorithm can benefit from $[m, \ell]$_SA objects when $k < \ell$.

5 Early Decision

This section extends the algorithm described in Figure 1 in order to obtain an early-deciding algorithm that allows the processes to decide by round $R_f = \min\big(\lfloor \frac{f}{\Delta} \rfloor + 2, \lfloor \frac{t}{\Delta} \rfloor + 1\big)$, where $\Delta = m\lfloor \frac{k}{\ell} \rfloor + (k \bmod \ell)$.

This algorithm is described in Figure 2 (its proof can be found in [21]). It is obtained from the base algorithm in a surprisingly simple way: only two new statements are added to the base algorithm to obtain early decision. These are the new lines, named A1 and A2, inserted between line 09 and line 10. No statement of the base algorithm has to be modified or suppressed.

The design principles of this algorithm are very simple. A process p_i that is a sender during a round r' and participates in the next round $r' + 1$ (so, it has not crashed by the end of r'), sends to all the processes a control message (denoted COMMIT) during the round $r' + 1$ (additional line A1). In that way, p_i informs all the processes that the estimate value it sent during the previous round r' was received by all the processes (this follows from the communication synchrony property). Moreover, as at most k different values are sent during a round (Lemma 2), and at least one process (namely, p_i) sent a value to all during r', it follows from the fact that p_i participates to the round $r' + 1$

that the estimates of all the processes contain at most k different values at the end of r'. Consequently, a process that receives a COMMIT message during a round $r'+1$ can decide the value of its estimate at the end of the round r' and stops (additional line A2).

It is easy to see that if at least one process in $p_1, \ldots, p_{\Delta}$ does not crash, the processes decide in two rounds. If all the processes $p_1, \ldots, p_{\Delta}$ crash and at least one process in $p_{\Delta+1}, \ldots, p_{2\Delta}$ does not crash, the decision is obtained in at most 3 rounds. Etc. It is interesting to observe that, when $m = \ell = k = 1$ we have $\Delta = 1$ and we obtain a remarkably simple uniform early deciding consensus algorithm for the classical round-based synchronous model $\mathcal{S}_{n,t}[\emptyset]$.

References

1. Aguilera, M.K., Toueg, S.: A Simple Bivalency Proof that t-Resilient Consensus Requires $t+1$ Rounds. Information Processing Letters 71, 155–178 (1999)
2. Aguilera, M.K., Le Lann, G., Toueg, S.: On the Impact of Fast failure Detectors on Real-Time Fault-Tolerant Systems. In: Malkhi, D. (ed.) DISC 2002. LNCS, vol. 2508, pp. 354–369. Springer, Heidelberg (2002)
3. Borowsky, E., Gafni, E.: Generalized FLP Impossibility Results for t-Resilient Asynchronous Computations. In: STOC 1993. Proc. 25th ACM Symposium on Theory of Distributed Computing, pp. 91–100. ACM Press, New York (1993)
4. Chaudhuri, S.: More Choices Allow More Faults: Set Consensus Problems in Totally Asynchronous Systems. Information and Computation 105, 132–158 (1993)
5. Chaudhuri, S., Herlihy, M., Lynch, N., Tuttle, M.: Tight Bounds for k-Set Agreement. Journal of the ACM 47(5), 912–943 (2000)
6. Dolev, D., Reischuk, R., Strong, R.: Early Stopping in Byzantine Agreement. Journal of the ACM 37(4), 720–741 (1990)
7. Fischer, M.J., Lynch, N.A.: A Lower Bound on the Time to Assure Interactive Consistency. Information Processing Letters 14(4), 183–186 (1982)
8. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of Distributed Consensus with One Faulty Process. Journal of the ACM 32(2), 374–382 (1985)
9. Gafni, E.: Round-by-round Fault Detectors: Unifying Synchrony and Asynchrony. In: PODC 2000. Proc. 17th ACM Symp. on Principles of Dist. Computing, pp. 143–152. ACM Press, New York (1998)
10. Gafni, E., Guerraoui, R., Pochon, B.: From a Static Impossibility to an Adaptive Lower Bound: The Complexity of Early Deciding Set Agreement. In: STOC 2005. Proc. 37th ACM Symposium on Theory of Computing, pp. 714–722. ACM Press, New York (2005)
11. Herlihy, M.P.: Wait-Free Synchronization. ACM TOPLAS 13(1), 124–149 (1991)
12. Herlihy, M.P., Penso, L.D.: Tight Bounds for k-Set Agreement with Limited Scope Accuracy Failure Detectors. Distributed Computing 18(2), 157–166 (2005)
13. Herlihy, M.P., Rajsbaum, S.: Algebraic Spans. MSCS 10(4), 549–573 (2000)
14. Herlihy, M.P., Shavit, N.: The Topological Structure of Asynchronous Computability. Journal of the ACM 46(6), 858–923 (1999)
15. Lamport, L., Fischer, M.: Byzantine Generals and Transaction Commit Protocols. Unpublished manuscript, pages 16 (April 1982)
16. Mostéfaoui, A., Rajsbaum, S., Raynal, M.: Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems. Journal of the ACM 50(6), 922–954 (2003)

17. Mostéfaoui, A., Rajsbaum, S., Raynal, M.: The Combined Power of Conditions and Failure Detectors to Solve Asynchronous Set Agreement. In: PODC 2005. Proc. 24th ACM Symposium on Principles of Distributed Computing, pp. 179–188. ACM Press, New York (2005)
18. Mostéfaoui, A., Rajsbaum, S., Raynal, M.: Synchronous Condition-Based Consensus. Distributed Computing 18(5), 325–343 (2006)
19. Mostéfaoui, A., Raynal, M.: k-Set Agreement with Limited Accuracy Failure Detectors. In: PODC 2000. 19th ACM Symp. on Principles of Distributed Computing, pp. 143–152 (2000)
20. Mostéfaoui, A., Raynal, M.: Randomized Set Agreement. In: SPAA 2001. Proc. 13th ACM Symposium on Parallel Algorithms and Architectures, pp. 291–297. ACM Press, New York (2001)
21. Mostéfaoui, A., Raynal, M., Travers, C.: Narrowing power vs efficiency in synchronous set agreement. Tech Report #1836, IRISA, Université de Rennes (France), pages 13 (2007)
22. Raynal, M., Travers, C.: Synchronous Set Agreement: A Concise Guided Tour (with open problems). In: PRDC 2006. Proc. 12th Int'l IEEE Pacific Rim Dependable Computing Symposium, pp. 267–274. IEEE Computer Press, Los Alamitos (2006)
23. Saks, M., Zaharoglou, F.: Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge. SIAM Journal on Computing 29(5), 1449–1483 (2000)