Parallel Consensus is Harder than Set Agreement in Message Passing

Zohir Bouzid
UPMC – LIP6-CNRS, France.
zohir.bouzid@lip6.fr

Corentin Travers
U. Bordeaux 1 – LaBRI-CNRS, France.
travers@labri.fr

***Abstract*—In the traditional consensus task, processes are required to agree on a common value chosen among the initial values of the participating processes. It is well known that consensus cannot be solved in crash-prone, asynchronous distributed systems. Two generalizations of the consensus tasks have been introduced: k-set agreement and k-parallel consensus.**

The k-set agreement task has the same requirements as consensus except that processes are allowed to decide up to k distinct values. In the k-parallel consensus task, each process participates simultaneously in k instances of consensus and is required to decide in at least one of them; any two processes deciding in the same instance must decide the same value.

It is known that both tasks are equivalent in the wait-free shared memory model. Perhaps surprisingly, this paper shows that this is no longer the case in the n-process asynchronous message passing model with at most t process crashes. Specifically, the paper establishes that for parameters t, n, k such that $t > \frac{n+k-2}{2}$, k-parallel consensus is strictly harder than k-set agreement.

The proof compares the information on failures necessary to solve each task in the failure detector framework and relies on a result in topological combinatorics, namely, the chromatic number of Kneser graphs. The paper also introduces the new failure detector class $V\Sigma_k$, which is a generalization of the quorum failures detector class Σ suited to k-parallel consensus.

***Keywords*-Consensus; Failure detectors; Fault tolerance; Agreement; Message passing; Kneser graph;**

I. Introduction

The k-set agreement problem: The k-set agreement problem (Chaudhuri, [1]) is one of the fundamental problems in fault tolerant distributed computing. Each process proposes a value and every non-faulty process is required to decide a value (termination) such that every decided value has been proposed (validity) and no more than k distinct values are decided (agreement). The problem generalizes the consensus problem, which corresponds to the case where $k = 1$. In asynchronous systems, it is well known that 1-set agreement is impossible as soon as at least one process may fail by crashing [2], whereas the case $k = n$ does not require any coordination at all. For intermediate values of k $(1 < k < n)$, asynchronous k-set agreement tolerating t process crash failures is possible if and only if $k > t$ [3], [4], [5], in shared memory or message-passing systems.

The k-parallel consensus problem: The k-parallel consensus problem [6][1] is another generalization of consensus. Each process proposes a value and is required to decide a pair (c, v), where c is an integer $1 \leq c \leq k$ and v is a proposed value (validity). Each non-faulty process has to decide (termination) and, for any two pairs with the same first component, the second component must be the same, i.e., for any decided pairs $(c, v), (c', v'), c = c' \implies v = v'$ (agreement). Intuitively, processes are trying to solve simultaneously k instances of the consensus problem in such a way that each process decides in at least one instance. For any instance, decisions occurring in that instance must be consistent with the validity and agreement requirements of consensus. A decided pair (c, v) may thus be interpreted as follows: value v is decided in the cth instance of consensus.

While k-set agreement weakens the safety property of consensus by allowing k values to be decided, k-parallel consensus may be though as weakening its liveness property by considering k instances in parallel, and allowing some instances to remain undecided. Practically, parallel consensus might be useful in situations where processes participate concurrently in k different applications: a k-parallel consensus protocol can guarantee progress in at least one application [9], [10].

Failure detectors: A failure detector is a distributed oracle that provides processes with possibly unreliable information on failures [11]. According to the quality of the information, several classes of failure detectors can be defined and may be used to solve otherwise impossible problems. For example, an eventual leadership failure detector (Ω, [12]) provides the processes with an id which is eventually 1) the same at each process and, 2) the id of a non-faulty process. Whereas k-set agreement cannot be solved in asynchronous message passing systems in which at most t processes may fail if $k \leq t$, [13] presents an asynchronous Ω-based k-set agreement protocol that tolerates up to t process failures, for any $t, k \leq t < \frac{kn}{k+1}$.

This work was supported in part by the DIGITEO project PACTOLE and by the ANR project DISPLEXITY.

[1]In this paper, k-parallel consensus problem is called k-simultaneous consensus. However, there is a body of work on simultaneous consensus, e.g. [7], [8], in which simultaneity refers to when agreement is reached. We use the terminology k-parallel consensus to avoid confusion.

A failure detector D is *necessary* for solving a distributed problem P if given any failure detector D' that can be used to solve P, it is possible to emulate D. It has been shown that a failure detector called Σ_k is necessary for k-set agreement in message passing systems [14].

k-set agreement vs. k-parallel consensus: The paper investigates the relative hardness of k-set agreement and k-parallel consensus in n-process asynchronous messages passing systems with crash failures. Let $t < n$ denote the bound on the number of failures.

Clearly, if a protocol for k-parallel consensus is provided, one can solve k-set agreement. A fundamental result in [6] is that the converse is true in asynchronous shared memory system. That is, both problems are computationally *equivalent* in shared memory: given any wait-free[2] protocol for k-set agreement (respectively, for k-parallel consensus), one can construct a wait-free protocol for k-parallel (respectively, for k-set agreement).

The equivalence has been instrumental in determining the weakest failure detector for k-set agreement in asynchronous shared memory systems [15], [16], a question which is still unsolved for message passing systems. In addition, while many k-set agreement protocols for messages passing systems with various synchrony assumptions or augmented with failure detectors have been proposed, e.g., [17], [18], [13], [19], [20], [21], [22], to the best of our knowledge no specific protocol is known for k-parallel consensus. A first step to remedy this situation will consist in a generic transformation for turning any k-set agreement protocol into a k-parallel consensus protocol, if such a transformation exists.

The equivalence extends to asynchronous process message passing systems when a majority of processes are non-faulty (i.e., $t < \frac{n}{2}$), as in this case shared memory can be emulated t-resiliently [23]. The question addressed in this paper is whether the equivalence between the two problems extends beyond the majority threshold. Our main result is that the answer is "no": we show that if $t > \frac{n+k-2}{2}$, k-parallel consensus is *strictly harder* than k-set agreement in an asynchronous n-process messages passing systems in which at most t processes can fail by crashing.

Contributions of the paper: We study both problems through the lens of the *amount of information on failures* required to solve them. This is usually captured in the framework of failure detectors. On one hand, it is known that failure detector $(\Sigma_k \times \Omega)$ is sufficient for k-set agreement [13]. On the other hand, we identify a new class of failure detectors, namely $V\Sigma_k$, and show that it is necessary for k-parallel consensus (Section IV). The question of whether k-parallel consensus can be solved t-resiliently using a k-set agreement protocol thus boils down to whether $V\Sigma_k$ can be emulated t-resiliently from $\Sigma_k \times \Omega$ (Section VI). If t is small enough, namely $t < \frac{kn}{k+1}$, Σ_k can be emulated t-resiliently without relying on any failure detector. In this case, it is enough to study for which values of t failure detector $V\Sigma_k$ can be implemented t-resiliently (Section V). It is shown that $\frac{n-k+2}{2}$ is a tight threshold. Interestingly, the proof relies on the chromatic number of a certain class of graphs, namely Kneser graphs. Finally, Section VII presents our main impossibility results, obtained by assembling the various pieces from the previous sections.

Table I summarizes the main contributions of the paper. k-SA and k-PC are shorthands for k-set agreement and k-parallel consensus respectively. $\mathcal{MP}_{n,t}$ denotes a message passing system made of n processes, t of which may crash. $\mathcal{MP}_{n,t}[\Omega]$ stands for system $\mathcal{MP}_{n,t}$ equipped with a failure detector Ω. $X \simeq X', X \prec X', X \preceq X'$ mean respectively that X and X' implements each other, X' implements X but X' does not implement X, X' implements X. See Section II for more details about the notations. Due to space limitations, some proofs are missing. They can be found in a companion technical report[24].

II. Preliminaries

Message passing asynchronous distributed system: We consider a distributed system made of a set Π of n asynchronous processes $\{p_1, \ldots, p_n\}$. Each process runs at its own speed, independently of the other processes.

Processes communicate by sending and receiving messages over a reliable but asynchronous network. Each pair of processes $\{p_i, p_j\}$ is connected by a bi-directional channel. Channels are reliable and asynchronous, meaning that each message sent by p_i to p_j is received by p_j after some finite, but unknown, time; there is no global upper bound on message transfer delays.

The system is equipped with a global clock whose ticks range $\mathbb{T}$ is the set of positive integers. This clock is not available to the processes, it is used from an external point of view to state and prove properties about executions.

Failures: Processes may fail by *crashing*. A process that crashes prematurely halts and never recovers. In an execution, a process is *faulty* if it fails and *correct* otherwise. A *failure pattern* is a function $\mathcal{F}$ from $\mathbb{T}$ to Π where $\mathcal{F}(\tau)$ is the set of processes that have failed by time τ. We define $Correct(\mathcal{F})$ and $Faulty(\mathcal{F}) = \Pi \backslash Correct(\mathcal{F})$ to be the set of correct processes and the set of faulty processes according to $\mathcal{F}$, respectively. When $\mathcal{F}$ is clear from the context, we simply write $Faulty$ and $Correct$ instead of $Faulty(\mathcal{F})$ and $Correct(\mathcal{F})$ respectively.

An *environment* (or *adversary* [25]) is a set of failure patterns. The *wait-free* environment consists in all failure patterns in which at least one process is correct. For $1 \leq t \leq n-1$, the *t-resilient* environment contains every failure

[2]A wait-free protocol tolerates any number of crash failures.

<table>
<tr><td>t</td><td colspan="2">0</td><td colspan="2">$\frac{n}{2}$</td><td colspan="2">$\frac{n+k-2}{2}$</td><td colspan="2">$\frac{kn}{k+1}$</td><td>n</td><td></td></tr>
<tr><td rowspan="5">$\mathcal{MP}_{n,t}$</td><td colspan="8">Σ_k implementable</td><td colspan="2">Σ_k not implementable</td><td>[14], Section V</td></tr>
<tr><td colspan="6">$V\Sigma_k$ implementable</td><td colspan="4">$V\Sigma_k$ not implementable</td><td>Section V</td></tr>
<tr><td colspan="6">$\Sigma_k \simeq V\Sigma_k$</td><td colspan="4">$\Sigma_k \prec V\Sigma_k$</td><td>Section VI</td></tr>
<tr><td colspan="4">k-SA $\simeq$ k-PC</td><td colspan="4">k-SA $\preceq$ k-PC</td><td colspan="4">k-SA $\prec$ k-PC</td><td>[6], Section VII</td></tr>
<tr><td colspan="8">k-PC solvable</td><td colspan="4">k-PC not solvable</td><td>Section VII</td></tr>
<tr><td>$\mathcal{MP}_{n,t}[\Omega]$</td><td colspan="10">$k$-SA solvable</td><td colspan="2">k-SA not solvable</td><td>[13]</td></tr>
</table>

Table I
CONTRIBUTIONS OF THE PAPER.

pattern in which no more than t processes are faulty (the $(n-1)$-resilient environment is the wait-free environment).

Failure detectors: Informally, a failure detector [11] is a distributed oracle that provides (perhaps inaccurate) hints on the current failure pattern of the execution. Operationally, a failure detector provides at each process p_i a read-only variable FD_i, whose value at time τ is denoted FD_i^τ. This value is the output of the failure detector for process p at time τ.

We recall next the main features of the framework in which failure detectors are defined, as introduced in [11]. A *failure detector history H with range $\mathcal{R}$* is a function $H : \Pi \times \mathbb{T} \rightarrow \mathcal{R}$. $H(p_i, \tau)$ may be seen as the output of the local failure detector module of process p_i at time τ. A *failure detector D with range $\mathcal{R}_D$* is a function that maps each failure pattern to set of failure detector histories with range $\mathcal{R}_D$. Given a failure pattern $\mathcal{F}$, $D(\mathcal{F})$ denotes the set of failure detector histories allowed by D when the failure pattern is $\mathcal{F}$.

For example, the range of the *quorum failure detector* Σ, defined in [26] is 2^Π, the set of all subsets of Π. $H : \Pi \times \mathbb{T} \rightarrow 2^\Pi \in \Sigma(\mathcal{F})$ iff $\forall \tau, \tau' \in \mathbb{T}, \forall p_i, p_j \in \Pi : H(p_i, \tau) \cap H(p_j, \tau') \neq \emptyset$ and $\exists \tau_c \in \mathbb{T} : \forall p_i \in Correct(F), \forall \tau \geq \tau_c, H(p_i, \tau) \subseteq Correct(\mathcal{F})$. That is, any two sets output by the failure detector intersect and eventually, for every correct process, the output of Σ contains only correct processes.

Comparing failure detectors: Let $D1, D2$ denote two failure detectors. Failure detector $D1$ is *weaker than* $D2$ in environment $\mathcal{E}$, denoted $D1 \preceq D2$, if there exists a distributed algorithm $\mathcal{T}_{D2 \rightarrow D1}$ that uses $D2$ to emulate the output of $D1$. More specifically, algorithm $\mathcal{T}_{D2 \rightarrow D1}$ maintains at each process p_i a variable OUT_{D1} intended to emulate the output of $D1$ at p_i; The variable can be used at each process to replace the actual output of $D1$: in any execution, p_i cannot distinguish between reading the variable OUT_{D1} or querying the failure detector $D1$. If $D1$ is weaker than $D2$ and $D2$ weaker than $D1$ in environment $\mathcal{E}$, $D1$ and $D2$ are said to be *equivalent* in $\mathcal{E}$ (denoted $D1 \simeq D2$). On the contrary, if $D2$ is not weaker than $D1$, $D1$ is *strictly weaker* than $D2$ (denoted $D1 \prec D2$).

Given a distributed task T, such as consensus, failure detector D is a *weakest failure detector* for T in environment $\mathcal{E}$ if (1) there exists an algorithm $\mathcal{A}_D$ for T in $\mathcal{E}$ that uses D and (2) for every failure detector D' that can be used to solve T in $\mathcal{E}$, there exists an algorithm $\mathcal{T}_{D' \rightarrow D}$ that uses D' to emulate D. Note that if $D1$ and $D2$ are weakest failure detectors for T, then $D1 \simeq D2$.

Comparing tasks: Given two distributed tasks $T1$ and $T2$ defined for n processes, task $T1$ *implements* task $T2$ in environment $\mathcal{E}$ if, given a protocol for $T1$, one can construct a protocol for $T2$ in $\mathcal{E}$ by interleaving steps of a message passing protocol with calls to any number of instances of the protocol for $T1$. The protocol for $T1$ is a "black-box": it is only required that it solves $T1$ in $\mathcal{E}$. We say that $T1$ is *harder* than $T2$ in $\mathcal{E}$ if $T1$ implements $T2$ whereas $T2$ does not implement $T1$.

Notations: Given var_i a local variable of process p_i, we denote by var_i^τ its value at time τ. $\mathcal{MP}_{n,t}$ denotes a n-process asynchronous message passing system in which at most t processes may fail. $\mathcal{MP}_{n,t}[D]$ denote the same system equipped with a failure detector of the class D. Given two failure detectors D, D' with ranges R_D and $R_{D'}$ respectively, $D \times D'$ denote the failure detector with range $R_D \times R_{D'}$ and histories $D(\mathcal{F}) \times D'(\mathcal{F})$ for any failure pattern $\mathcal{F}$.

III. THE FAILURE DETECTORS Σ_k, $V\Sigma_k$ AND Ω

This section recalls the definition of the failure detector classes Σ_k, Ω and introduces the new class $V\Sigma_k$. For each process p_i , FD_i^τ denotes the value output by the failure detector at time τ.

The family $\{\Sigma_k\}_{1 \leq k \leq n}$: A failure detector of the class Σ_k maintains at each process a variable QUORUM_i that contains at any time a set of process ids. The sets output, called *quorums*, satisfy the following properties:

- *Intersection*. Any set containing at least $k+1$ quorums has two intersecting quorums. Formally, let Q be the set of *all* quorums output at all the processes at all times. That is, $Q = \{B \mid \exists p_i \in \Pi, \exists \tau \in \mathbb{T} : \text{QUORUM}_i^\tau = B\}$. Then, for every $X \subseteq Q$ with $|X| > k : \exists B, B' \in X : B \cap B' \neq \emptyset$.
- *Liveness*. Eventually, for each correct process, every quorum contains only correct processes ids. That is, $\exists \tau \in \mathbb{T} : \forall p_i \in Correct, \forall \tau' \geq \tau : quorum_i^{\tau'} \subseteq Correct$

Σ_k was introduced in [14] where it is shown to be necessary to solve k-set agreement in message passing systems. A

$(\Sigma_k \times \Omega)$-based protocol tolerating any number of process crashes that solves k-set agreement is presented in [13]. The class Σ_1 is the same as Σ, the weakest failure detector to implement a register in crash-prone message passing systems [26].

The eventual leader failure detector Ω: A failure detector of the class Ω maintains at each process p_i a variable LEADER_i that contains a process id. It satisfies the following property:

- *Eventual leadership*. Eventually, for every correct process p_i, LEADER_i contains forever the same identity of a correct process. That is, $\exists p_\ell \in \mathit{Correct}, \exists \tau \in \mathbb{T} : \forall \tau' \geq \tau, \forall p_i \in \mathit{Correct}, \text{LEADER}_i^{\tau'} = \ell$.

$(\Omega \times \Sigma)$ is the weakest failure detector for consensus in message passing systems in any environment [12], [26].

The family $\{V\Sigma_k\}_{1 \leq k \leq n}$: The failure detector $V\Sigma_k$ (read Vector-Σ_k) outputs at each process p_i an array QUORUMS_i of size k. At any time, each component $\text{QUORUMS}_i[c], 1 \leq c \leq k$ of the array contains a set of process ids (a quorum). Intuitively, a failure detector $V\Sigma_k$ may be seen as k instances of a failure detector of the class Σ. In each instance, the intersection property of the class Σ is satisfied, whereas the liveness property may not hold. It is only required that liveness is satisfied in at least one instance. Formally:

- *Intersection*. Any two quorums output in the same entry c at the same process or at distinct processes intersect. That is, $\forall c, 1 \leq c \leq k, \forall \tau, \tau' \in \mathbb{T}, \forall p_i, p_{i'} \in \Pi : \text{QUORUMS}_i^{\tau}[c] \cap \text{QUORUMS}_{i'}^{\tau'}[c] \neq \emptyset$.
- *Liveness*. There exists some entry c such that, eventually, every quorum output in this entry at any correct process contains only correct processes. That is, $\exists \tau \in \mathbb{T}, \exists c, 1 \leq c \leq k : \forall p_i \in \mathit{Correct}, \forall \tau' \geq \tau, \text{QUORUMS}_i^{\tau'}[c] \subseteq \mathit{Correct}$.

IV. Necessity of $V\Sigma_k$ for k-parallel consensus

This section shows that failure detector $V\Sigma_k$ is necessary for solving k-parallel consensus. That is, failure detector $\mathcal{D}$ can be used to solve k-parallel consensus, then $V\Sigma_k$ can be emulated using $\mathcal{D}$.

Theorem IV.1. *For all t, k, $1 \leq t, k \leq n$, for any protocol $\mathcal{A}$ and any failure detector $\mathcal{D}$, if $\mathcal{A}$ solves k-parallel consensus in $\mathcal{MP}_{n,t}[\mathcal{D}]$ then $V\Sigma_k \preceq \mathcal{D}$.*

The strategy of the proof is similar to the one in [14]. There, it is shown that failure detector Σ_k is necessary to solve k-set agreement in message passing systems. The proof is simple and elegant, and, as we are about to see, can be generalized to the case of k-parallel consensus.

A protocol that emulates a failure detector $V\Sigma_k$ is described in Figure 1. Recall that we are given an algorithm $\mathcal{A}$ and a failure detector $\mathcal{D}$ such that $\mathcal{A}$ solves k-parallel consensus in $\mathcal{MP}_{n,t}[\mathcal{D}]$. We assign for each set $S \in 2^\Pi$ an instance of $\mathcal{A}$ denoted $\mathcal{A}^S$. Each process p_i participates in instance $\mathcal{A}^S$ only if $p_i \in S$. The value proposed by p_i in this instance is $\langle S, i \rangle$. In more details, algorithm $\mathcal{A}$ consists in n automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, one per process. Process p_i starts 2^{n-1} copies of $\mathcal{A}_i$, one copy, denoted $\mathcal{A}_i^S$, for each set $S \in 2^\Pi : i \in S$. The proposal of p_i in $\mathcal{A}_i^S$ is $\langle S, i \rangle$ and each message sent by $\mathcal{A}_i^S$ is tagged for the purpose of not confusing messages sent in different instances of $\mathcal{A}$. When a receive step is performed in $\mathcal{A}_i^S$, a message with tag S (if any) is selected from p_i's input message buffer and delivered to the automata. Failure detector queries are performed normally: when a failure detector value is needed by $\mathcal{A}_i^S$, the local failure detector module of p_i is queried. p_i performs steps of each automata $\mathcal{A}_i^S, i \in S$ in any fair way, for example in a round-robin fashion.

The output of the failure detector $V\Sigma_k$ at process p_i consists in a k-component array OUT_i. Process p_i maintains in addition a k-component array of sets denoted Q_i. If p decides (c, d) in instance $\mathcal{A}^S$, set S is added to the cth component of $Q_i[c]$.

For each $c, 1 \leq c \leq k$, p_i periodically strives to assign to the cth component of OUT_i a set $S \in Q_i[c]$ that contains only correct process (if $Q_i[c]$ contains such a set). To that end, each process periodically broadcasts HEARTBEAT messages (task T2). HEARTBEATs are used to rank processes id. Process p_i maintains an ordered list $order_i$ of processes ids. Each time a HEARTBEAT from process p_j is received, j is moved at the beginning of the list (Task T3). As each faulty process sends finitely many HEARTBEATs, there exists a time after which each correct process id appears before any faulty process id. Therefore, given two sets S, S' of process, $S \subseteq \mathit{Correct}$ and $S' \not\subseteq \mathit{Correct}$, the largest rank of the ids of the processes in S is eventually always smaller than the largest rank of the ids in S'. When p_i updates $\text{OUT}_i[c]$, it selects a set S that has the smallest largest rank of its ids among the sets currently in $Q_i[c]$ output by $V\Sigma_k$ (lines 5–7). If $Q_i[c]$ contains a set of correct processes, this guarantees that eventually $\text{OUT}_i[c] \subseteq \mathit{Correct}$.

Process p_i may not decide in every instance $\mathcal{A}^S$ in which it participate, as it may wait forever for messages from some process $p_j, j \notin S$. However, for any S such that $\mathit{Correct} \subseteq S$, every correct process that participates in $\mathcal{A}^S$, must eventually decide since $\mathcal{A}$ correctly solves k-parallel consensus. Hence, some component c_i of Q_i eventually contains a set of correct processes, and therefore, $\text{OUT}_i[c_i]$ is eventually a subset of the correct processes. Moreover, the protocol ensures that when a set S is added to $Q_i[c_i]$, it is eventually added to the c_ith component of every Q_j, for every correct process p_j (tasks T1 and T4). It thus follows that eventually at each correct process p_j, $\text{OUT}_j[c] \subseteq \mathit{Correct}$ for some $c, 1 \leq c \leq k$, thereby ensuring the liveness property of the class $V\Sigma_k$.

For the intersection property of the class $V\Sigma_k$, it remains to see that for any two sets S, S' assigned to the cth

component of the emulated failure detector output, perhaps at different processes, $S \cap S' \neq \emptyset$, for any $c, 1 \leq c \leq k$. Note that S, S' are assigned to the cth component of the emulated failure detector only if (c, d) and (c, d') are decided in instances $\mathcal{A}^S$ and $\mathcal{A}^{S'}$ respectively.

Let S, S' be two disjoint subsets of Π. Suppose that processes $p \in S$ and $p' \in S'$ decide the pairs (c, d) and (c', d') in $\mathcal{A}^S$ and $\mathcal{A}^{S'}$ respectively. We demonstrate that, in this case, $c \neq c'$. First, as any value proposed in $\mathcal{A}^S$ (respectively, in $\mathcal{A}^{S'}$) is of the form $\langle S, j\rangle$ (respectively, $\langle S', j\rangle$), where $p_j \in S$ (respectively, $p_j \in S'$) $d \neq d'$ by the validity of property of k-parallel consensus. Second, because $S' \cap S = \emptyset$, prefixes of the executions of $\mathcal{A}^S$ and $\mathcal{A}^{S'}$ can be "merged" in a single execution of $\mathcal{A}$ in which the set of participating processes is $S \cup S'$. That is, there exists a single execution α of $\mathcal{A}$ that is indistinguishable from the execution of $\mathcal{A}^S$ (respectively, of $\mathcal{A}^{S'}$) for p (respectively, for p'). p and p' thus decide respectively (c, d) and (c', d') in α. As $d \neq d'$, it thus follows that $c' \neq c$. A proof can be found in the full version [24].

init $Q_i[1..k] \leftarrow [\{\Pi\}, \ldots, \{\Pi\}]$; /* array of set of sets */
$order_i \leftarrow (1, \ldots, n)$; /* ordered list of processes ids */
OUT$[1..k] \leftarrow [\Pi, \ldots, \Pi]$; /* $V\Sigma_k$ output */
for each $S \in 2^\Pi : i \in S$ **do**
launch an instance $\mathcal{A}_i^S$ of $\mathcal{A}_i$ with input $\langle S, i\rangle$ **enddo**
/* instances run in parallel independently */
start tasks T1,T2,T3,T4

task T1: **when** p_i decides in $\mathcal{A}_i^S$:
(1) **let** (c, d) be the decision of p_i; $Q_i[c] \leftarrow Q_i[c] \cup \{S\}$;
(2) *send* (c, S) to *all* /* $d = \langle S, j\rangle$ for some $j \in S$ */

task T2: **repeat periodically**
(3) *send* HEARTBEAT(i) *to* all

task T3: **when** HEARTBEAT(j) is received:
(4) move j at the head of the list $order_i$
(5) **for each** $c : 1 \leq c \leq k$ **do** OUT$_i[c] \leftarrow E$
(6) where $E \in Q_i[c]$ and $\forall S \in Q_i[c]$,
(7) $\max_{j \in E} rank(j, order_i) \leq \max_{j \in S} rank(j, order_i)$
/* $rank(j, order_i)$: rank of j in the ordered list $order_i$ */

task T4: **when** (c, S) is received:
(8) $Q_i[c] \leftarrow Q_i[c] \cup \{S\}$;

Figure 1. Emulation of $V\Sigma_k$ from an algorithm $\mathcal{A}$ that uses a failure detector $\mathcal{D}$ to solve k-parallel consensus (code for p_i)

V. t-RESILIENT PROTOCOLS FOR $V\Sigma_k$ AND Σ_k

This section investigates whether there is a t-resilient protocol for implementing a failure detector Σ_k or $V\Sigma_k$. For the class Σ_k, the answer is known [13]. For completeness, the result is recalled at the end of this section (Theorem V.5).

For the class $V\Sigma_k$, we show that the existence of a protocol emulating a failure detector $V\Sigma_k$ is strongly related to the chromatic number of a certain family of graphs, namely, Kneser graphs. We show that there exists a t-resilient protocol that emulates a failure detector $V\Sigma_k$ if and only if the Kneser graph $KG_{n,n-t}$ has a proper vertex k-coloring.

A. t-resilient emulation of $V\Sigma_k$

This section is devoted to the proof of the following Theorem:

Theorem V.1. *Let n, k, t be integers such that $1 \leq t, k \leq n$. There exists a protocol that emulates a failure detector of the class $V\Sigma_k$ in $\mathcal{MP}_{n,t}$ if and only if $t \leq \frac{n+k-2}{2}$.*

Preliminaries: A *coloring* of a graph is a labelling of the graph's vertices with colors drawn from the integers $\{1, 2, 3, \ldots\}$. A coloring is called a *k-coloring* if it uses at most k colors and *proper* if no two adjacent vertices share the same color. The *chromatic number* of a graph G, denoted $\chi(G)$, is the smallest number of colors needed to properly color G, *i.e.* the smallest value of k for which a proper k-coloring of G is possible.

The *Kneser graph* $KG_{n,k}$ is the undirected graph whose vertices are the subsets of k elements of a set of size n, and where two vertices share an edge whenever the two corresponding sets are disjoint. For example, $KG_{n,1}$ is the complete graph, and $KG_{5,2}$ is isomorphic to the Petersen graph. An important result about Kneser graphs is their chromatic number. The chromatic number $\chi(KG_{n,k})$ of the Kneser graph is exactly $n - 2k + 2$ if $n \geq 2k$, and 1 otherwise. This result was conjectured by Martin Kneser early in 1955 and proved for the first time by Lovász[27] in 1978. His proof was the first one using algebraic topology to solve a problem in combinatorics, giving rise to the field of topological combinatorics. Simpler proofs were later given by Bárány [28], Greene [29] and Matoušek [30].

Kneser graphs, and their chromatic number are central to show that it is impossible to emulate t-resiliently $V\Sigma_k$ for certain values of n, k and t. We reduce the existence of a protocol that emulates $V\Sigma_k$ in $\mathcal{MP}_{n,t}$ to the problem of whether k colors are sufficient to properly color $KG_{n,n-t}$ (Lemma V.2). Conversely, we show that if $V\Sigma_k$ can be emulated in $\mathcal{MP}_{n,t}$, there is a k-coloring of $KG_{n,n-t}$ (Lemma V.3).

A t-resilient protocol emulating $V\Sigma_k$: A simple algorithm that emulates a failure detector of the class $V\Sigma_k$ in $\mathcal{MP}_{n,t}$ is described in Figure 2. The algorithm requires that $t \leq \frac{n+k-2}{2}$.

The algorithm relies on a k-coloring of the Kneser graph $KG_{n,n-t}$. As seen in the preliminaries, the chromatic number of $KG_{n,n-t}$ is $\chi(KG_{n,n-t}) = n - 2(n-t) + 2 = 2t - n + 2 \leq k$ if $n \geq 2(n-t)$ and 1 otherwise. Therefore, for $t \leq \frac{n+k-2}{2}$, the graph $KG_{n,n-t}$ can be properly colored with k colors.

The processes are initially provided with a function *color* that maps each subset of Π of size $n-t$ to an integer in the range $[1,k]$ such that any two disjoint sets are mapped to distinct integers. That is, *color* is a k-coloring of $KG_{n,n-t}$. Since the chromatic number of this graph is $\chi(KG_{n,n-t}) = 2t-n+2$ if $n \geq 2(n-t)$ and 1 otherwise, the function *color* does exist as $t \leq \frac{n+k-2}{2}$.

Each process p_i maintains a vector of sets of processes $\text{OUT}_i[1..k]$ intended to contain the output of the emulated failure detector $V\Sigma_k$. Each entry of OUT_i is initially equal to Π, the set of processes ids in the system. To ensure the liveness property of $V\Sigma_k$, i.e., the existence for each correct process p_i of an entry ℓ_i such that eventually, the ℓ_ith entry of OUT_i contains only correct processes ids, each process periodically broadcasts HEARTBEAT (line 1). When a HEARTBEAT is received from some process p_j, the identity of p_j is added to the local set Q_i (line 2). Whenever $n-t$ distinct ids have been accumulated in Q_i, the output of $V\Sigma_k$ is updated as follows. The current set S of ids in Q_i is assigned to the cth entry of OUT_i, where c is the color of S (line 3). Then p_i broadcasts Q_i together with its color c in a QUORUM message. Finally, Q_i is reset to the empty set. When a QUORUM(Q_j, c) message is received from p_j, the cth element of OUT_i is updated to Q_i.

Note that it thus follows that eventually, some entry of the output of $V\Sigma_k$ at all processes contains only correct processes, as every faulty process eventually stops sending HEARTBEATs messages, and thus its id eventually stops occurring in Q_i. Moreover, using the map *color* to assign sets of $(n-t)$ processes ids to entries of the vector OUT_i guarantees the intersection property of $V\Sigma_k$. Indeed, by definition of the map *color*, two sets are assigned to the same entry only if they intersect.

init $\text{OUT}_i[1..k] \leftarrow [\Pi, \ldots, \Pi]$; /* output of $V\Sigma_k$ */
color : $\{S \subseteq \Pi : |S| = n-t\} \rightarrow \{1, \ldots, k\}$; /* k-coloring of $KG_{n,n-t}$ */
$Q_i \leftarrow \emptyset$; /* set of processes ids, initially empty */
start task T1,T2,T3

task T1: **repeat** periodically
(1) *send* HEARTBEAT(i) to *all*

task T2: **when** HEARTBEAT(j) is received:
(2) $Q_i \leftarrow Q_i \cup \{j\}$;
(3) **if** $|Q_i| = n-t$ **then let** $c = color(Q_i)$; $\text{OUT}_i[c] \leftarrow Q_i$;
(4) *send* QUORUM(Q_i, c) to *all*; $Q_i \leftarrow \emptyset$ **endif**

task T3: **when** QUORUM(Q, c) is received:
(5) $\text{OUT}_i[c] \leftarrow Q$

Figure 2. Emulation of $V\Sigma_k$ in $\mathcal{MP}_{n,t}$, $t \leq \frac{n+k-2}{2}$ (code for p_i)

Lemma V.2. *Let n, t, k be integers such that $1 \leq t, k \leq n$ and $t \leq \frac{n+k-2}{2}$. The algorithm described in Figure 2 implements a failure detector $V\Sigma_k$ in $\mathcal{MP}_{n,t}$.*

Proof: The proof is divided in two parts corresponding to the two properties of the class $V\Sigma_k$, namely liveness and intersection.

Liveness. Define τ to be the time at which: (i) all faulty processes have already crashed, (ii) all messages sent by them have already been received by correct processes. There exists a time $\tau' \geq \tau$ such that for every correct process p_i, the processes that are accumulated in Q_i at any time after τ' are correct. This is because there are at least $n-t$ correct processes, and each of them never stops sending HEARTBEAT messages (line 1). Let $Q \subseteq$ *Correct* be any set of $n-t$ ids accumulated by a given correct process p_ℓ after τ', say at τ_ℓ. Let c denote $color(Q)$. Every process $p_j \in$ *Correct* $\setminus \{p_\ell\}$ will receive the message QUORUM(c, Q) from p_ℓ, say at τ_j. Hence for every $p_i \in$ *Correct*, $\text{OUT}_i[c]$ contains Q at τ_i. After τ_i, since $\tau_i \geq \tau'$, whether $\text{OUT}_i[c]$ is updated (as a result of accumulating a new set of ids by p_i or receiving a QUORUM message) or not, its value is necessarily a subset of *Correct*.

Intersection. Let $\ell, 1 \leq \ell \leq k$ and let S_i, S_j be two sets of processes ids corresponding to the ℓth entry of the output of the emulated failure detector at some processes p_i and p_j (with p_i not necessarily distinct from p_j). That is, at some time τ_i (respectively, τ_j), $\text{OUT}_i[\ell] = S_i$ (respectively, $\text{OUT}_j[\ell] = S_j$). If S_i or S_j is equal to Π, $S_i \cap S_j \neq \emptyset$ as $\text{OUT}_i[\ell]$ and $\text{OUT}_j[\ell]$ always contain processes ids. Otherwise, S_i is the value of the variable Q_i at some time, and ℓ is the color assigned to the set S_i by the map *color*. Similarly, *color* maps S_j to ℓ. Since disjoint sets are mapped to distinct colors, it follows that $S_i \cap S_j \neq \emptyset$. ■

An impossibility result: We now prove that the condition linking t, k and n of Lemma V.2 is tight for the existence of a protocol that emulates $V\Sigma_k$ in $\mathcal{MP}_{n,t}$:

Lemma V.3. *Let n, t, k be integers such that $1 \leq t, k \leq n$ and $\frac{n+k-2}{2} < t$. There is no algorithm that emulates a failure detector of the class $V\Sigma_k$ in $\mathcal{MP}_{n,t}$.*

We actually prove this Lemma as a corollary of a slightly more general result:

Lemma V.4. *Let n, t, k be integers such that $1 \leq t, k \leq n$ and $\frac{n+k-2}{2} < t$. There is no algorithm that emulates a failure detector of the class $V\Sigma_k$ in $\mathcal{MP}_{n,t}[\Omega]$.*

Proof: The proof is by contradiction. Assume that there exists an algorithm $\mathcal{A}$ that implements a failure detector $V\Sigma_k$ in $\mathcal{MP}_{n,t}[\Omega]$ with $\frac{n+k-2}{2} < t$, i.e, $k < 2t-n+2$. We show that we can use $\mathcal{A}$ to properly color $KG_{n,n-t}$ with k colors. This contradicts the fact that the chromatic number of $KG_{n,n-t}$ is $\chi(KG_{n,n-t}) = 2t-n+2$ when $t \geq \frac{n}{2}$ and 1 otherwise.

Let $\mathcal{S} = S_1, \ldots, S_u, u = \binom{n}{n-t}$ be an enumeration of all subsets of Π of size $n-t$. We construct an execution α of $\mathcal{A}$ from which we derive a proper k-coloring of $KG_{n,n-t}$. The construction proceeds inductively by forming longer and

longer prefix α_i of α. At the end of α_i, each set $S_j, 1 \leq j \leq i$ has received a color $c_j, 1 \leq c_j \leq k$ such that any two disjoint sets receive distinct colors.

Base step. Let α'_1 be an execution of $\mathcal{A}$ in which the set of correct processes is S_1. Moreover, the faulty processes are initially crashed in α'_1. At each process $p_i \in S_1$, the output of the failure detector Ω is the same process id ℓ_1, where $p_{\ell_1} \in S_1$. Let p_j be a process in S_1. By the liveness property of $V\Sigma_k$, there exists a time τ_1 and an entry c_1 such that, at time τ_1, the c_1th entry of the failure detector output at process p_j is a set $S \subseteq S_1$. This is because S_1 is the set of correct processes in α'_1 and eventually one entry of the vector output by $V\Sigma_k$ at each correct process must contain only correct processes ids.

In execution α_1, no process fails. However, processes in $\Pi \backslash S_1$ do not take a step before time τ_1. Moreover, execution α_1 and α'_1 are indistinguishable up to time τ_1 for every process in S_1. In particular, for every process in S_1, the output of Ω until time τ_1 is ℓ_1. Hence, as in execution α'_1, process $p_j \in S_1$ output at time τ_1 a vector whose c_1th entry is a set contained in S_1. We then let every process take enough steps for every message sent before τ_1 to be received. The color c_1 is assigned to S_1.

Induction step. Suppose that the prefix α_i has been constructed, for some $i, 1 \leq i < u$. We describe how to extend α_i to form the prefix α_{i+1}.

Let α'_{i+1} be an execution of $\mathcal{A}$ that extends α_i and in which the set of correct processes is S_{i+1}. More precisely, α_i is a prefix of α'_{i+1} and every process in $\Pi \setminus S_{i+1}$ fails immediately after α_i. Processes in S_{i+1} then keep taking steps forever and, after α_i, the output of Ω at each process in S_{i+1} is the same id ℓ_{i+1} for some process $p_{\ell_{i+1}} \in S_{i+1}$. The eventual leadership property of the class Ω is thus satisfied. Let p_j be an arbitrary process in S_{i+1}. As in the base case, it follows from the liveness property of the class $V\Sigma_k$ that there exists an entry c_{i+1} such that eventually the c_{i+1}th entry of the vector output by $\mathcal{A}$ at p_j is included in S_{i+1}, which is the set of correct processes in that execution. Let τ_{i+1} be a time following α_i at which this occurs.

Execution α_{i+1} and α'_{i+1} are indistinguishable for every process in S_{i+1} up to time τ_{i+1}. In particular, for every process in S_{i+1}, the output of Ω is ℓ_{i+1} after α_i and until time τ_{i+1}, and processes in $\Pi \backslash S_{i+1}$ take no step after α_i and until τ_{i+1}. After τ_{i+1}, we let each process takes enough step in order to every message sent before τ_{i+1} to be received. As α_{i+1} and α'_{i+1} are indistinguishable for every process in S_{i+1}, the c_{i+1}th entry of the vector output by $\mathcal{A}$ at process p_j at time τ_{i+1} is the same as in α'_{i+1}, that is a set included in S_{i+1}. We assign the color c_{i+1} to S_{i+1}.

Final step. Suppose we have constructed prefix α_u as described above. Execution α is an infinite execution with prefix α_u. After α_u, each process takes infinitely many steps, for example in round-robin fashion; the output of Ω at each process is the same id of some arbitrary correct process. Every message sent is eventually received.

Note that execution α is a valid execution of $\mathcal{A}$ in $\mathcal{MP}_{n,t}[\Omega]$ with no failure. In particular, note that since after prefix α_u, the underlying failure detector outputs the same correct process id at every process, the failure detector history is a valid history for a failure detector of the class Ω. The output of $\mathcal{A}$ must therefore fulfill the properties of the class $V\Sigma_k$. We claim that the coloring of each $S_i, 1 \leq i \leq u$ with $c_i, 1 \leq i \leq u$, as indicated in the construction is a proper k-coloring of $KG_{n,n-t}$.

Notice first that each c_i is an entry of the vector of size k output by $\mathcal{A}$, i.e., $1 \leq c_i \leq k$. Finally, let S_i, S_j be two sets such that $S_i \cap S_j = \emptyset$. By construction, at time τ_i, the c_ith entry of the vector output by $\mathcal{A}$ at some process is a set $s_i \subseteq S_i$. Similarly, at time τ_j, the c_jth entry of the vector output by $\mathcal{A}$ at some process is a set $s_j \subseteq S_j$. As $S_i \cap S_j = \emptyset$, we have $s_i \cap s_j = \emptyset$. It thus follows from the intersection property of $V\Sigma_k$ that $c_i \neq c_j$, as desired.

As $1 \leq k$, and $\frac{n+k-2}{2} < t$, it follows that $2(n-t) \leq n$. The chromatic number of $KG_{n,n-t}$ is thus $\chi(KG_{n,n-t}) = n - 2(n-t) + 2 = 2t - n + 2$. This is a contradiction since we have $k < 2t - n + 2$. ∎

Lemma V.3 is a consequence of Lemma V.4, as any protocol emulating $V\Sigma_k$ in $\mathcal{MP}_{n,t}$ emulates $V\Sigma_k$ in any system in which a failure detector is available. Theorem V.1 then immediately follows from Lemma V.2 and Lemma V.3.

B. t-resilient emulation of Σ_k

For completeness, we recall here the condition linking the parameters t, k and n under which there exists a t-resilient protocol emulating a failure detector Σ_k:

Theorem V.5 ([13]). *Let n, k, t be integers such that $1 \leq t, k \leq n$. There exists a protocol that emulates a failure detector Σ_k in $\mathcal{MP}_{n,t}$ if and only if $t < \frac{kn}{k+1}$.*

VI. $\{V\Sigma_k\}_{1 \leq k \leq n}$ vs. $\{\Sigma_k\}_{1 \leq k \leq n}$

This section compares the two families $\{V\Sigma_k\}_{1 \leq k \leq n}$ and $\{\Sigma_k\}_{1 \leq k \leq n}$. For establishing that k-set agreement is weaker than k-parallel consensus, we are mainly interested in the values of parameters n, t, k for which $V\Sigma_k$ can be emulated in $\mathcal{MP}_{n,t}[\Sigma_k]$. This is because Σ_k can be used to solve k-set agreement and $V\Sigma_k$ is necessary for k-parallel consensus. Hence, a protocol that uses a k-set agreement protocol to solve k-parallel consensus in $\mathcal{MP}_{n,t}$ implies that $V\Sigma_k$ can be emulated in $\mathcal{MP}_{n,t}[\Sigma_k]$. Nevertheless, for completeness, we also study for which values of the parameters n, t and k a failure detector Σ_k can be emulated in $\mathcal{MP}_{n,t}[V\Sigma_k]$ (Section VI-B).

A. Emulation of $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[\Sigma_k]$

The values of t, n, k and k' for which a failure detector $V\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}[\Sigma_k]$ are completely characterized by the following theorem:

Theorem VI.1. *Let $n, t, k, k', 1 \leq k, k', t < n$. There is a protocol that emulates a failure detector $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[\Sigma_k]$ if and only if $k = 1$ or $t \leq \frac{n+k'-2}{2}$.*

Proof: If $t \leq \frac{n+k'-2}{2}$, we know from Lemma V.2 that there is a protocol that emulates $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}$, and thus also in $\mathcal{MP}_{n,t}[\Sigma_k]$. If $k = 1$, $V\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}[\Sigma]$ by simply replicating k' times the outputs of Σ.

Suppose now that $k \geq 2$ and $t > \frac{n+k'-2}{2}$. Assume for contradiction that there exists a protocol $\mathcal{A}$ that emulates $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[\Sigma_k]$.

Let $S_1, \ldots, S_u, u = \binom{n}{n-t}$ be every subset of Π of size $n - t$. For every $S_i \in \mathcal{S}$, let α_i be an execution of $\mathcal{A}$ in which (1) the set of correct processes is S_i, (2) each faulty process fails initially and (3) the output of the underlying failure detector Σ_k is always S_i, at every process. By the liveness property of the class $V\Sigma_{k'}$, there exists a time τ_i and $c_i, 1 \leq c_i \leq k'$ such that the c_ith entry of the vector output by $V\Sigma_{k'}$ at some correct process in α_i is a set $s_i \subseteq S_i$.

Now, we show that there exists S_j, S_ℓ such that $S_i \cap S_j = \emptyset$ and $c_j = c_\ell = c$. Assume not for contradiction. This means that $f : S_i \mapsto c_i$ is a k'-coloring of $KG_{n,n-t}$. Therefore, the chromatic number of $KG_{n,n-t}$ is $\chi(KG_{n,n-t}) \leq k'$. As $t > \frac{n+k'-2}{2}$, it follows that $k' < 2t - n + 2$. Contradiction!

Let α be an execution of $\mathcal{A}$ defined as follows. The set of correct processes in α is $S_j \cup S_\ell$. At each process in S_j (respectively, S_ℓ), the output of Σ_k is always S_j (respectively, S_ℓ). Since $k \geq 2$, this is a valid output for Σ_k in an execution where the set of correct processes is $S_j \cup S_\ell$. Faulty processes do not take a step in α. As for correct processes, each message sent by the processes in S_j (respectively, S_ℓ) before time $\tau = \max(\tau_j, \tau_\ell)$ is delayed if it is sent to a process in S_ℓ (respectively, S_j). Otherwise it is received as in α_j (respectively, α_ℓ). After time τ, every delayed message is received, and every message sent after that time is eventually received.

Up to time τ, α and α_j are thus indistinguishable for any processes in S_j and, similarly, α and α_ℓ are indistinguishable for any processes in S_ℓ. Therefore, in α, the cth($= c_j = c_\ell$) entry of the vector output by $\mathcal{A}$ is a set $s_j \subseteq S_j$ at some process, and a set $s_\ell \subseteq S_\ell$ at some other process. As $S_j \cap S_i = \emptyset$, this contradicts the intersection property of the class $V\Sigma_{k'}$. ∎

The impossibility part of the Theorem can be extended to the case in which a failure detector Ω is available:

Corollary VI.2. *Let $n, t, k, k', 1 \leq k, k', t < n$. There is no protocol that emulates a failure detector $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[\Sigma_k \times \Omega]$ if $k \geq 2$ and $t > \frac{n+k'-2}{2}$.*

Proof: Essentially the same strategy as in the previous proof can be reused. What is left undefined in the executions α_i considered there is the output of the failure detector Ω. In each $\alpha_i, 1 \leq i \leq u$, Ω may always outputs the same process $q \in S_i$ at each process. Then, in the definition of α, the output of Ω is the same as in α_j (respectively, α_ℓ) up to time τ for each process in S_j (respectively, S_ℓ). After time τ, the output of Ω is the same process $\in S_i \cup S_\ell$ at every correct process in α. This is consistent with the eventual leadership property of the class Ω and does not help each process in S_j (respectively, S_ℓ) to distinguish until time τ between α_j (respectively, α_ℓ) and α. ∎

B. From $V\Sigma_k$ to Σ_k

Next theorem complements Theorem VI.1 by characterizing the values of the parameters t, n and k' for which it is possible to t-resiliently emulate $\Sigma_{k'}$ when a failure detector $V\Sigma_k$ is available. The proof can be found in the full version [24].

Theorem VI.3. *Let $n, t, k, k', 1 \leq k, k', t < n$. There is a protocol that emulates a failure detector $\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[V\Sigma_k]$ if and only if $k \leq k'$ or $t < \frac{k'n}{k'+1}$.*

VII. Separation results

This section glues together the results presented in Section IV, Section V and Section VI to establish the following main theorem:

Theorem VII.1. *Let $n, t, k, k', 1 \leq k, k' \leq t < n$ such that $2 \leq k$ and $\frac{n+k'-2}{2} < t$. There is no protocol for k'-parallel consensus in $\mathcal{MP}_{n,t}[k\text{-SA}]$.*

The proof strategy is as follows: On one hand, it has been shown in [13] that k-set agreement can be solved in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$, for any value of t. On the other hand, we have seen that $V\Sigma_{k'}$ is necessary for solving k'-parallel consensus (Theorem IV.1). The impossibility of a t-resilient solution to k'-parallel consensus using a k-set agreement protocol thus reduces to the impossibility of a t-resilient emulation of $V\Sigma_{k'}$ based on $\Omega \times \Sigma_k$. The latter has been answered in the previous Section (Corollary VI.2).

Proof: Let $k \geq 2$ and $t > \frac{n+k'-2}{2}$. The proof is by contradiction. Assume that there exists a k'-parallel consensus protocol $\mathcal{A}$ in $\mathcal{MP}_{n,t}[k\text{-SA}]$. More precisely, $\mathcal{A}$ uses any number of copies of a k-set agreement protocol $\mathcal{B}$ to solve k'-parallel consensus. Protocol $\mathcal{B}$ is any t-resilient protocol for k-set agreement. No assumption is made regarding the internals of protocol $\mathcal{B}$. It particular, $\mathcal{B}$ might be a failure detector-based protocol.

It is known that k-set agreement can be solved in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$ [13] – the protocol presented there imposes no requirement on t and k. $\mathcal{B}$ may thus be the protocol presented in [13]. Therefore, by combining protocol $\mathcal{A}$ and $\mathcal{B}$, it follows that k'-parallel consensus can be solved in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$.

In section IV, we have shown that $V\Sigma_k$ is necessary for k'-parallel consensus. That is, if there is a k'-parallel consensus protocol using a failure detector $\mathcal{D}$, then one may use $\mathcal{D}$ to emulate a failure detector of the class $V\Sigma_k$. As k'-parallel consensus can be solved in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$ by

combining protocols $\mathcal{A}$ and $\mathcal{B}$, it thus follows that there exists a protocol $\mathcal{T}$ that emulates $V\Sigma_k$ in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$.

However, by corollary VI.2, there is no protocol that emulates $V\Sigma_k$ in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$ if $k \geq 2$ and $t > \frac{n+k'-2}{2}$: a contradiction. ■

The relative hardness of k-set agreement and k-parallel consensus is also expressed by the following theorem. The theorem gives tight bounds on t for k-set agreement and k-parallel consensus to be solvable when an eventual leader is available:

Theorem VII.2. *Let $1 \leq k \leq t < n$. In $\mathcal{MP}_{n,t}[\Omega]$,*

1) *There is a k-set agreement protocol if and only if $t < \frac{kn}{k+1}$;*
2) *There is a k-parallel consensus protocol if and only if $t \leq \frac{n+k-2}{2}$.*

Proof: 1) Bonnet and Raynal[14] have shown that Σ_k is necessary for k-set agreement. Moreover, Σ_k can be emulated in $\mathcal{MP}_{n,t}[\Omega]$ if and only if $t < \frac{kn}{k+1}$. Hence, the existence of an Ω-based k-set agreement protocol tolerating t implies a t-resilient emulation of Σ_k in $\mathcal{MP}_{n,t}[\Omega]$. This is not possible if $t \geq \frac{kn}{k+1}$. A k-set agreement protocol in $\mathcal{MP}_{n,t}[\Sigma_k \times \Omega]$ is presented in [13]. Since Σ_k can be emulated in $\mathcal{MP}_{n,t}$ provided that $t < \frac{kn}{k+1}$, k-set agreement can be solved in $\mathcal{MP}_{n,t}[\Omega]$ for $t < \frac{kn}{k+1}$.

2) Similarly, we have shown that $V\Sigma_k$ is necessary for k-parallel (Section IV) and that $V\Sigma_k$ can be emulated in $\mathcal{MP}_{n,t}[\Omega]$ if and only if $t \leq \frac{n+k-2}{2}$ (Lemma V.4). Therefore, no protocol solves k-parallel consensus in $\mathcal{MP}_{n,t}[\Omega]$ if $t > \frac{n+k-2}{2}$.

For $t \leq \frac{n+k-2}{2}$, k-parallel consensus can be solved in $\mathcal{MP}_{n,t}[\Omega]$ as follows: Each process p_i participates parallelly in k instances $\mathcal{A}^1, \ldots, \mathcal{A}^k$ of an $(\Omega \times \Sigma)$-based consensus protocol $\mathcal{A}$. p_i proposes its initial value in each instance; if p_i decides v in instance $\mathcal{A}^c$, the pair (c, v) is returned by p_i as its output for k-parallel consensus. Emulation of Σ in instance j consists in outputting the jth component of the vector provided by $V\Sigma_k$.

The liveness property of the class $V\Sigma_k$ ensures that for some $c, 1 \leq c \leq k$, the cth entry of the vector output by $V\Sigma_k$ eventually contains a subset of the correct processes. Moreover, for any entry j, every pair of sets in the jth entry of the vector provided by $V\Sigma_k$ have a non-empty intersection. Therefore, the cth entry of the output of $V\Sigma_k$ satisfy the same property as the output of a failure detector Σ. It thus follows that every correct process eventually decides in $\mathcal{A}^c$.

Since in every instance $\mathcal{A}^j, 1 \leq j \leq k$, the emulation of Σ preserves the intersection property, no two process decide different values in $\mathcal{A}^j$. This is because the safety property of consensus relies only on the fact that the intersection of any two sets output by Σ is non-empty. Hence, if (j, v) and (j, v') are decided, then $v = v'$, for any $j, 1 \leq j \leq k$. ■

VIII. Conclusion

The paper has investigated the relationship linking the k-set agreement and the k-parallel consensus problems in asynchronous crash-prone message-passing systems. While k-parallel consensus t-resiliently implements k'-set agreement for any value of $t < n$, and any $k' \leq k$, the paper has shown that k-set agreement does not implement k'-parallel consensus for any $k', 1 \leq k' \leq n$ if $\frac{n+k'-2}{2} < t$ and $2 \leq k$. In addition, when an eventual leader Ω is available, it has established that k-parallel consensus can be solved if and only if $t \leq \frac{n+k-2}{2}$. This is to be compared with k-set agreement, which can be solved if and only if $t < \frac{kn}{k+1}$.

The paper leaves open the following question: For $\frac{n}{2} \leq t \leq \frac{n+k-2}{2}$, what is the smallest value of k' for which k-set agreement t-resiliently implements k'-parallel consensus in message passing? As a starting point, we demonstrate below that given a k-set agreement protocol and assuming $t \leq \frac{n+k-2}{2}$, $(2k-1)$-parallel consensus can be achieved. Note that this is not trivial: for example, if $2(2k+1) < n$, $2k - 1 \leq t$ for every t, $\frac{n}{2} \leq t \leq \frac{k+n-2}{2}$.

Partition the set of processes in two sets $A = \{p_1, \ldots, p_{k-1}\}$ and $B = \{p_k, \ldots, p_n\}$. Each p_i in A broadcasts (i, v_i) where v_i is p_i's initial value before deciding that pair. The $n - k + 1$ remaining processes emulate a shared memory reduction of k-parallel consensus to k-set agreement, as described in [6], assuming that no more than $t_B = \lfloor \frac{n-k}{2} \rfloor$ processes among them fail. If $p_i \in A$ obtains a decision (c, d), it broadcast the pair $(c+(k-1), d)$ before deciding it. Any process that receives a pair (c, d) decides this pair. Note that we have $1 \leq c \leq 2k-1$, and if pairs $(c, d), (c, d')$ are decided, $d = d'$. If the number of actual failures in B exceeds t_B, the emulation may not terminate, but safety is not violated (that is, a simulated write or read operation may not terminate).

For termination, let f_A, f_B be the actual number of failures in sets A and B respectively. If $f_A < k - 1$, the protocol terminates as A contains a correct process. Otherwise, $f_A = k-1$ and consequently $f_B \leq t-(k-1) \leq \frac{n+k-2}{2} - (k-1) = \frac{n-k}{2}$.That is, a majority of the processes in A are correct. The emulation of the shared memory protocol thus terminates, which enables every correct process to decide.

Another avenue for future research is to determine the weakest failure detector for k-parallel consensus. A candidate is the (new) failure detector class $\mathcal{W}_k$: the output of $\mathcal{W}_k$ is a k-component array. Each component contains the same output as a failure detector $\Omega \times \Sigma$, except that the eventual leadership property of Ω and the liveness property of Σ are only required to hold in a single component. This component must be the same for both properties and for every process. See [24] for more details.

REFERENCES

[1] S. Chaudhuri, "More choices allow more faults: set consensus problems in totally asynchronous systems," *Inf. Comput.*, vol. 105, no. 1, pp. 132–158, 1993.

[2] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[3] E. Borowsky and E. Gafni, "Generalized flp impossibility result for t-resilient asynchronous computations," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. ACM, 1993, pp. 91–100.

[4] M. Herlihy and N. Shavit, "The topological structure of asynchronous computability," *J. ACM*, vol. 46, no. 6, pp. 858–923, 1999.

[5] M. E. Saks and F. Zaharoglou, "Wait-free k-set agreement is impossible: The topology of public knowledge," *SIAM J. Comput.*, vol. 29, no. 5, pp. 1449–1483, 2000.

[6] Y. Afek, E. Gafni, S. Rajsbaum, M. Raynal, and C. Travers, "The *k*-simultaneous consensus problem," *Distributed Computing*, vol. 22, no. 3, pp. 185–195, 2010.

[7] B. A. Coan and C. Dwork, "Simultaneity is harder than agreement," *Inf. Comput.*, vol. 91, no. 2, pp. 205–231, 1991.

[8] Y. Moses and M. R. Tuttle, "Programming simultaneous actions using common knowledge," *Algorithmica*, vol. 3, pp. 121–169, 1988.

[9] E. Gafni and R. Guerraoui, "Generalized universality," in *Proceedings 22nd International Conference on Concurrency Theory (CONCUR)*, ser. Lecture Notes in Computer Science, vol. 6901. Springer, 2011, pp. 17–27.

[10] E. Gafni, S. Rajsbaum, M. Raynal, and C. Travers, "The committee decision problem," in *Proceedings 7th Latin American Symposium on Theoretical Informatics (LATIN)*, ser. Lecture Notes in Computer Science, vol. 3887. Springer, 2006, pp. 502–514.

[11] T. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, pp. 225–267, 1996.

[12] T. Chandra, V. Hadzilacos, and S. Toueg, "The weakest failure detector for solving consensus," *J. ACM*, vol. 43, no. 4, pp. 685–722, 1996.

[13] Z. Bouzid and C. Travers, "(anti-omega^x $\times$ σ_z-based k-set agreement algoritms," in *Proceedings 14th International Conference on Principles of Distributed Systems (OPODIS)*, ser. Lecture Notes in Computer Science, vol. 6490. Springer, 2010, pp. 189–204.

[14] F. Bonnet and M. Raynal, "On the road to the weakest failure detector for k-set agreement in message-passing systems," *Theor. Comput. Sci.*, vol. 412, no. 33, pp. 4273–4284, 2011.

[15] E. Gafni and P. Kuznetsov, "The weakest failure detector for solving k-set agreement," in *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 2009, pp. 83–91.

[16] P. Zielinski, "Anti-*omega*: the weakest failure detector for set agreement," *Distributed Computing*, vol. 22, no. 5-6, pp. 335–348, 2010.

[17] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg, "Partial synchrony based on set timeliness," *Distributed Computing*, vol. 25, no. 3, pp. 249–260, 2012.

[18] D. Alistarh, S. Gilbert, R. Guerraoui, and C. Travers, "Of choices, failures and asynchrony: The many faces of set agreement," *Algorithmica*, vol. 62, no. 1-2, pp. 595–629, 2012.

[19] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and A. Tielmann, "The weakest failure detector for message passing set-agreement," in *Proceedings 22nd International Symposium on Distributed Computing (DISC)*, ser. Lecture Notes in Computer Science, vol. 5218. Springer, 2008.

[20] A. Mostéfaoui, S. Rajsbaum, M. Raynal, and C. Travers, "On the computability power and the robustness of set agreement-oriented failure detector classes," *Distributed Computing*, vol. 21, no. 3, pp. 201–222, 2008.

[21] A. Mostéfaoui, M. Raynal, and J. Stainer, "Relations linking failure detectors associated with k-set agreement in message-passing systems," in *13th Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, ser. Lecture Notes in Computer Science, vol. 6976. Springer, 2011, pp. 341–355.

[22] P. R. Parvédy, M. Raynal, and C. Travers, "Strongly terminating early-stopping *k*-set agreement in synchronous systems with general omission failures," *Theory Comput. Syst.*, vol. 47, no. 1, pp. 259–287, 2010.

[23] H. Attiya, A. Bar-Noy, and D. Dolev, "Sharing memory robustly in message-passing systems," *J. ACM*, vol. 42, no. 1, pp. 124–142, 1995.

[24] Z. Bouzid and C. Travers, "Simultaneous Consensus is Harder than Set Agreement in Message Passing," full version. [Online]. Available: http://hal.inria.fr/hal-00752610

[25] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and A. Tielmann, "The disagreement power of an adversary," *Distributed Computing*, vol. 24, no. 3-4, pp. 137–147, 2011.

[26] C. Delporte-Gallet, H. Fauconnier, and R. Guerraoui, "Tight failure detection bounds on atomic object implementations," *J. ACM*, vol. 57, no. 4, 2010.

[27] L. Lovász, "Kneser's conjecture, chromatic number, and homotopy," *Journal of Combinatorial Theory, Series A*, vol. 25, no. 3, pp. 319–324, 1978.

[28] J. Bárány, "A short proof of kneser's conjecture," *J. of Combinatorial Theory, Series A*, vol. 25, no. 3, pp. 325–326, 1978.

[29] J. Greene, "A new short proof of kneser's conjecture," *The American mathematical monthly*, vol. 109, no. 10, pp. 918–920, 2002.

[30] J. Matoušek, "A combinatorial proof of knesers conjecture," *Combinatorica*, vol. 24, no. 1, pp. 163–170, 2004.