Simultaneous Consensus is Harder than Set Agreement in Message Passing

Zohir Bouzid, Corentin Travers

Simultaneous 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

November 16, 2012

Abstract

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

Keywords: Consensus, Failure detectors, Fault tolerance, Agreement, Message passing, Kneser graph.

1 Introduction

The k-set agreement problem The k-set agreement problem (Chaudhuri, [1]) is one of the fundamental problem 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 impossible if and only if $k > t$ [3, 4, 5], in shared memory or message-passing system.

The k-simultaneous consensus problem The k-set agreement problem [6] 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 $(c, v), (c', v'), c = c' \Rightarrow v = v'$ (agreement). Intuitively, processes are trying to solve simultaneously k instances of the consensus problem in such 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-simultaneous consensus may be though as weakening the liveness property of consensus by considering k instances in parallel, and allowing some instances to remain undecided. Practically, simultaneous consensus might be useful in situations where processes participate concurrently in k different applications: a k-simultaneous consensus protocol can guarantee progress in at least one application [7, 8].

Failure detectors A failure detector is a distributed oracle that provides processes with possibly unreliable information on failures [9]. 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, [10]) provides the processes with an id which is eventually 1) the same at each process, 2) the id of a non-faulty process. Whereas k-set agreement cannot be solved if $k \leq t$, an Ω-based k-set agreement protocol is presented in [11], for $k < t < \frac{kn}{k+1}$.

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.

k-set agreement vs. k-simultaneous consensus The paper investigates the relative hardness of k-set agreement and k-simultaneous 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-simultaneous 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[1] protocol for k-set agreement (respectively, for k-simultaneous consensus), one can construct a wait-free protocol for k-simultaneous (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 [12, 13], 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 has been proposed, e.g., [14, 15, 11, 16, 17, 18, 19], to the best of our knowledge no specific protocol is known for k-simultaneous consensus. A first step to remedy this situation will consist in a generic transformation for turning any k-set agreement protocol into a k-simultaneous protocol, it 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[20]. The question addressed in the 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-simultaneous consensus is *strictly harder* than k-set agreement

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

in a 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 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 [11]. On the other hand, we identify a new class of failure detector, namely Σ_k, and show that it is necessary for k-simultaneous consensus (Section 4). The question of whether k-simultaneous 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 6). 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 $V\Sigma_k$ can be implemented t-resiliently (Section 5). 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 7 presents our main impossibility results, obtained by assembling the various pieces from the previous section. Table 1 summarizes the main contributions of the paper (k-SA and k-SC are shorthands for k-set agreement and k-simultaneous consensus respectively). $\mathcal{MP}_{n,t}$ denotes a message passing system made of n processes, t of which may crash. $\mathcal{MP}_{n,t}[\Omega]$ is 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 2 for more details about the notations.

<table>
<tr><td>t</td><td colspan="4">0 $\quad\quad$ $\frac{n}{2}$ $\quad\quad$ $\frac{n+k-2}{2}$ $\quad\quad$ $\frac{kn}{k+1}$ $\quad\quad$ n</td><td></td></tr>
<tr><td rowspan="5">$\mathcal{MP}_{n,t}$</td><td colspan="3">Σ_k implementable</td><td>Σ_k not implementable</td><td>[21], Section 5</td></tr>
<tr><td colspan="2">$V\Sigma_k$ implementable</td><td colspan="2">$V\Sigma_k$ not implementable</td><td>Section 5</td></tr>
<tr><td colspan="2">$\Sigma_k \simeq V\Sigma_k$</td><td colspan="2">$\Sigma_k \prec V\Sigma_k$</td><td>Section 6</td></tr>
<tr><td>k-SA $\simeq$ k-SC</td><td>k-SA $\preceq$ k-SC</td><td colspan="2">k-SA $\prec$ k-SC</td><td>[6], Section 7</td></tr>
<tr><td colspan="2">k-SC solvable</td><td colspan="2">k-SC not solvable</td><td>Section 7</td></tr>
<tr><td>$\mathcal{MP}_{n,t}[\Omega]$</td><td colspan="3">$k$-SA solvable</td><td>$k$-SA not solvable</td><td>[11]</td></tr>
</table>

Table 1: Contributions of the paper.

2 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 messages transfers 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 $Faulty(\mathcal{F}) = \Pi \setminus Correct(\mathcal{F})$ 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 *Correct* and *Fauly* instead of *Faulty*($\mathcal{F}$) and *Correct*($\mathcal{F}$) respectively.

An *environment* (or *adversary* [22]) is a set of failure patterns. The *wait-free* environment consists in all failure pattern in which at least one process is correct. For $1 \leq t \leq n-1$, the *t-resilient* environment contains every failure 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 [9] 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 [9]. A *failure detector history H with range* $\mathcal{R}$ is a function $H : \Pi \times \mathbb{T} \to \mathcal{R}$. $H(p_i, \tau)$ 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 [23] is 2^Π, the set of all subsets of Π. $H : \Pi \times \mathbb{T} \to 2^\Pi \in \Sigma(\mathcal{F})$ iff $\forall \tau, \tau' \in \mathbb{T}, \forall p_i, p_j \in \Pi : H(p_i, \tau) \cap H(p_i, \tau')$ 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 process.

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 $\mathcal{T}_{D2 \to D1}$ that uses $D2$ to emulate the output of $D1$. More specifically, algorithm $\mathcal{T}_{D2 \to D1}$ 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 $\mathcal{T}_{D' \to D}$ that uses D' to solve D. Note that if $D1$ and $D2$ are weakest failure detector 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 $\mathcal{T} \in$ in $\mathcal{E}$ by interleaving steps of a message protocol with calls to any number of instance of the protocol $T1$. 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 $T2$ is $\mathcal{E}$ 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. $\mathcal{MP}_{n,t}$ denote 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 rand R_D and $R_{D'}$, $D \times D'$ denote the failure detector with range $R_D \times R_{D'}$ and histories $D(\mathcal{F}) \times D'(\mathcal{F})$ for any failure pattern $\mathcal{F}$.

3 The failure detectors classes 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 denote the 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 processes 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 processes, the each quorums 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 [21] 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 crashed that solves k-set agreement is presented in [11]. The class Σ_1 is the same as Σ, the weakest failure detector to implement a register in crash-prone message passing systems [23].

The eventual leader failure detector Ω A failure detector of the class Ω maintains at each process $p_i a$ 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, $\exists p_\ell \in Correct, \exists \tau \in \mathbb{T} : \forall \tau' \geq \tau, \forall p_i \in Correct, \text{LEADER}_i^{\tau'} = \ell$.

$(\Omega \times \Sigma)$ is the weakest failure detector for consensus in message passing systems in any environment [10, 23].

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, $\exists \tau \in \mathbb{T}, \exists c, 1 \leq c \leq k : \forall p_i \in Correct, \forall \tau' \geq \tau, \text{QUORUMS}_i^{\tau'}[c] \subseteq Correct$.

4 Necessity of $V\Sigma_k$ for k-simultaneous consensus

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

Theorem 4.1. *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-simultaneous 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 [21]. There, it is shown that failure detector Σ_k is necessary to solve k-set agreement in message passing systems. The proof is simple an elegant, and, as we are about to see, can be generalized to the case of k-simultaneous 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-simultaneous 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 $\mathcal{A}_i^S$, per 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 in $\mathcal{A}_i^S$ is tagged for the purpose of not confusing messages sent in different instances of $\mathcal{A}$. For example, whenever $\mathcal{A}_i^S$ produces a message m, S is appended to m before it is sent over the underlying network. 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 an k-components 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 HEARTBEAT messages, 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 4–4). 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 $S, \mathit{Correct} \subseteq S$, every correct process that participates in $\mathcal{A}^S$, must eventually decide since $\mathcal{A}$ correctly solves k-simultaneous 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 process. 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 of Q_j, for every correct process p_j (task 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 observe that, in this case, $c \neq c'$.

First, as the values proposed by each process p_j that participates in $\mathcal{A}^S$ (respectively, in $\mathcal{A}^{S'}$) is $\langle S, j\rangle$ (respectively, $\langle S', j\rangle$), $d \neq d'$ by the validity of property of k-simultaneous 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 $c' \neq c$.

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$[1..k] \leftarrow [\Pi, \ldots, \Pi]$; `/* emulated failure detector output */`
for each $S \in 2^\Pi : i \in S$ **do** launch an instance $\mathcal{A}_i^S$ of $\mathcal{A}_i$ with input $\langle S, i\rangle$ **end do**
`/* 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; $Q_i[c] \leftarrow Q_i[c] \cup \{S\}$; *send* (c, S) to *all* `/*` $d = \langle S, j\rangle$ `for some` $j \in S$ `*/`

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

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

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

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

Proof The proof is essentially the same as the proof of Bonnet and Raynal [24]. In order for the paper to be self-contained, a complete proof is provided next.

If var_i a variable local to process p_i, let var_i^τ denote the value of the variable at time τ. We consider an arbitrary infinite execution α of the protocol described in Figure 1. For each process p_i, the output of the emulated failure detector $V\Sigma_k$ is stored in the array OUT$_i$; OUT$_i^\tau[c]$ thus denotes the cth quorum output at process p_i by the emulated failure detector at time τ, for any $c, 1 \leq c \leq k$, and any $p_i \in \Pi$. The proof is divided in two parts. We first establish that the protocol ensures the Liveness property of the class $V\Sigma_k$ (Lemma 4.2) and then show that the Intersection property is also ensured (Lemma 4.3). The correctness of the emulation then immediately follows (Theorem 4.1).

Lemma 4.2 (Liveness)**.** *For each correct process p_i, there exists a time τ_i and $\ell_i, 1 \leq \ell_i \leq k$: $\forall \tau \geq \tau_i$, OUT$_i^\tau[\ell_i] \subseteq$ Correct.*

Proof. Let p_i denote a correct process in α. Recall that $\mathcal{A}$ is an asynchronous algorithm that solves the k-simultaneous consensus problem in $\mathcal{MP}_{n,t}[\mathcal{D}]$. This implies that every correct process eventually decides in every instance $\mathcal{A}^S$, where $Correct \subseteq S$. In particular, p_i decides in the instance $\mathcal{A}^{Correct}$. Let (ℓ_i, v_i) be the pair eventually decided by p_i in this instance. It thus follows from the protocol (task T1), that, after some time $\tau 1$, $Correct \in Q_i[\ell_i]$.

If no set containing faulty process is ever inserted in $Q_i[\ell_i]$, then $\text{OUT}_i^\tau[\ell_i]$ is a set of the correct processes in α, for any time τ. Otherwise, let S be any set containing a faulty process that is included in $Q_i[\ell_i]$. That is, after some time $\tau 2$, we have $S \in Q_i^\tau[\ell_i]$, for any $\tau \geq \tau 2$. Let p_f denote a faulty process in S. p_f sends finitely many HEARTBEAT messages to p_i. Therefore, p_f is moved finitely many times at the head of the list $order_i$. On the contrary, p_i receives infinitely many HEARTBEAT messages from each correct process, and thus each correct process is moved infinitely often at the head of the list $order_i$. It thus follows that after some time τ_3, we always have

$$\max_{p_j \in Correct} rank(j, order_i) < \max_{p_j \in S} rank(j, order_i)$$

Hence, after time $\max(\tau_1, \tau_2, \tau_3)$, the set stored in $\text{OUT}_i[\ell_i]$ cannot be S (lines 4–8, task T3). As this is true for any set S containing a faulty process, $\text{OUT}_i[\ell_i]$ eventually always contain a set of correct processes. □

Lemma 4.3. *For any p_i, p_j pair of (not necessarily distinct) processes, for any times τ, τ', for any $c, 1 \leq c \leq k$:* $\text{OUT}_i^\tau[c] \cap \text{OUT}_j^{\tau'}[c] \neq \emptyset$

Proof. Let S, S' be the values of $\text{OUT}_i[c]$ and $\text{OUT}_j[c]$ at times τ and τ' respectively. Assume for contradiction that $S \cap S' = \emptyset$. Note that $S, S' \neq \Pi$ and thus neither S nor S' are the initially values of $\text{OUT}_i[c]$ and $\text{OUT}_j[c]$ respectively. Therefore, by the code (Task T3), S must have been inserted in $Q_i[c]$ by p_i and S' inserted in $Q_j[c]$ by p_j. This implies that p_i has decided (c, v) in the instance $\mathcal{A}^S$ and p_j has decided (c, v') in $\mathcal{A}^{S'}$, where v, v' are the cth entry of a vector proposed in $\mathcal{A}^S$ and $\mathcal{A}^{S'}$ respectively.

Any value proposed in $\mathcal{A}^S$ has the form $\langle S, \ell \rangle$, where $p_\ell \in S$. Similarly, any value proposed in $\mathcal{A}^{S'}$ has the form $\langle S', \ell' \rangle$, where $p_{\ell'} \in S'$. Hence, by the validity requirement of the k-simultaneous problem and the fact that $S \neq S'$, $v \neq v'$.

We next construct an execution β of $\mathcal{A}$ in which p_i decides (c, v) and p_j decides (c, v'). Since $v \neq v'$, this is a contradiction. Intuitively, β is obtained by merging steps of $\mathcal{A}^S$ and $\mathcal{A}^{S'}$ taken by the processes in $S \cup S'$ in α in a single execution.

More precisely, the failure pattern is the same in β and α. The failure detector history of the underlying failure detector $\mathcal{D}$ is also the same in α and β. In β, only processes in $S \cup S'$ take steps. Let σ (respectively, σ') be the sequence of steps taken by the processes in S (resp. S') while executing $\mathcal{A}^S$ (resp., $\mathcal{A}^{S'}$) in α. The sequence of steps μ taken by the processes in $S \cup S'$ in β is obtained by merging σ and σ' respecting the real order occurrence of the steps in α. Note that β is a valid execution of $\mathcal{A}$, as the history of $\mathcal{D}$ is the same in α and β.

Since p_i (resp., p_j) takes the same steps in β and in the execution of $\mathcal{A}^S$ (resp., $\mathcal{A}^{S'}$) in α, it decides (c, v) (resp., (c, v')) in β. This is a contradiction as $v \neq v'$ and $\mathcal{A}$ solves the k-simultaneous problem in $\mathcal{MP}_{n,t}[\mathcal{D}]$. □

It thus follows from Lemma 4.2 and Lemma 4.3 that the protocol of Figure 1 emulates failure detector of the class $V\Sigma_k$ given a protocol $\mathcal{A}$ using a failure detector $\mathcal{D}$ to solve k-simultaneous consensus.

5 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 from previous work [21, 11]. For completeness, the result is recalled at the end of this section (Section 5.2, Theorem 5.5).

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, the Kneser graphs. For any integer $x < n$, the vertices of the Kneser graph $KG_{n,x}$ are the subset of size x of $\{1, \ldots, n\}$; any two subsets share an edge if and only if their intersection is empty. 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 k-coloring.

5.1 t-resilient emulation of $V\Sigma_k$

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

Theorem 5.1. *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 problem of finding the chromatic number of general graphs is NP-complete [25].

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[26] 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 [27], Greene [28] and Matoušek [29].

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 5.2). 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 5.3).

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 the Kneser graph $KG_{n,n-t}$. 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 an 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). Q_i is then reset to the empty set.

Note that it thus follows that eventually, some entry of the output of $V\Sigma_k$ at process p_i 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 $\text{OUT}_i[1..k] \leftarrow [\Pi, \ldots, \Pi]$; `/* emulated failure detector output */`
color : $(n-t)$-elements subsets of $\Pi \longrightarrow \{1, \ldots, k\}$; `/* 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) **for each** $p_j \in \Pi$ **do** *send* HEARTBEAT(i) to p_j **enddo**

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

task T4: **when** $V\Sigma_k$ is queried:
(4) return OUT_i

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

A detailed proof of the protocol follows.

Lemma 5.2. *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. here "local" liveness is proved

- Liveness. Let p_i denote a correct process. Let us observe that infinitely many times a "fresh" set of $n-t$ ids is accumulated in Q_i (line 3). This is because there are at least $n-t$ correct processes, and each of them never stops sending HEARTBEAT messages (line 1). Since faulty processes eventually stop sending HEARTBEAT messages, there is a time after which only HEARTBEAT from correct processes are received by p_i. Hence, Q_i eventually contains only ids of corect processes. Therefore, there exists an entry ℓ_i and time after which we have $\text{OUT}_i[\ell_i] \subseteq \mathit{Correct}$.

- 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 5.2 is tight for the existence of a protocol that emulates $V\Sigma_k$ in $\mathcal{MP}_{n,t}$:

Lemma 5.3. *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 5.4. *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 ids 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 $\Pi \setminus S_1$ 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 keeps 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 $p_{\ell_{i+1}} \in S_{i+1}$. 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 $\Pi \setminus S_{i+1}$ take no step after α_i and until τ_{i+1}. We then 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 arbitrary process id. 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 output 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 fulfills 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 $2(n-t) \leq n$. Therefore, the chromatic number of $KG_{n,n-t}$ is $\chi(KG_{n,n-t}) = n - 2(n-t) + 2 = 2t - n + 2$. This is a contradiction since $k < 2t - n + 2$. □

Lemma 5.3 is a consequence of Lemma 5.4, as any t-resilient protocol emulating $V\Sigma_k$ would also emulate $V\Sigma_k$ in an environment in which a failure detector is available. Theorem 5.1 then immediately follows from Lemma 5.2 and Lemma 5.3.

5.2 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 5.5 ([11]). *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 $t < \frac{kn}{k+1}$.*

Proof. The proof considers two cases, according to the value of t.

- $t < \frac{kn}{k+1}$. A simple protocol to emulate a failure detector of the class Σ_k in $\mathcal{MP}_{n,t}$ is as follows: Every process periodically broadcasts HEARTBEAT messages. Whenever a process has collected $n-t$ "fresh" HEARTBEAT from a set S of $n-t$ distinct processes, it updates the output of Σ_k with this set S. The intersection property of the class Σ_k is ensured since, as $(k+1)(n-t) > n$, among any $k+1$ sets of size $n-t$ of processes, at least two sets intersect. Liveness follows from the fact that faulty processes eventually stop broadcasting heartbeats.

- $t \geq \frac{kn}{k+1}$. In that case, it is not possible to emulate Σ_k in $\mathcal{MP}_{n,t}$. The proof is by contradiction and relies on a partitioning argument.

  Assume for contradiction that there exists an algorithm $\mathcal{A}$ that emulates a failure detector Σ_k in $\mathcal{MP}_{n,t}$. We construct an execution α of $\mathcal{A}$ in which the intersection property of the class Σ_k is violated.

As $t \geq \frac{kn}{k+1}$, i.e., $(k+1)(n-t) \leq n$, there exists a collection $\mathcal{S}$ of $k+1$ pairwise disjoint subsets of Π of size $n-t$, $\mathcal{S} = \{S_1, \ldots, S_{k+1}\}$. That is, for all $1 \leq i \neq j \leq k+1$, $S_i \cap S_j = \emptyset$ and $|S_i| = |S_j| = n-t$. We construct inductively longer an longer prefix α_i of α.

- Base case. Let α'_1 denote an execution of $\mathcal{A}$ in which every process, but the $n-t$ processes of S_1 initially fail. By the liveness property of the class Σ_k, there exists a time τ_1 at which the output of Σ_k is a set $s_1 \subseteq S_1$ at some process $p_j \in S_1$.

  In the prefix α_1, no process fails. Every process in $\Pi \setminus S_1$ does not take any step before τ_1 and, for every process in S_1, α_1 and α'_1 are indistinguishable up to time τ_1. Hence, as in α'_1, the output of Σ_k at process $p_j \in S_1$ is $s_1 \subseteq S_1$. After τ_1, we then let every process take enough steps such that every message sent before τ_1 is received, and every process in $\Pi \setminus S_1$ takes at least one step.
- Induction base. Suppose that the prefix α_i has been constructed, for some $i, 1 \leq i < k+1$. 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, after α_i, every process in $\Pi \setminus S_{i+1}$ fails and every process in S_{i+1} keeps taking steps forever. By the liveness property of the class Σ_k, there exists a time τ_{i+1} after α_i at which the output of Σ_k is some set $s_{i+1} \subseteq S_{i+1}$ at some process $p_j \in S_{i+1}$.

  Execution α_{i+1} and α'_{i+1} are indistinguishable for every process until τ_{i+1}. As in the base case, after time τ_{i+1}, we let every process takes enough steps for every message sent before τ_{i+1} to be delivered. Note that, as in α'_{i+1}, the output of Σ_k at process $p_j \in S_{i+1}$ is $s_{i+1} \subseteq S_{i+1}$ at time τ_{i+1} in execution α_{i+1}.
- Final case. Suppose that prefix α_{k+1} has been constructed. α is an infinite execution of $\mathcal{A}$ with no faulty process. After α_{k+1}, every process takes infinitely many steps in some arbitrary order in such a way that every message sent is eventually received.

α is a valid execution of $\mathcal{A}$ in $\mathcal{MP}_{n,t}$ in which the emulated failure detector output sets $s_1, \ldots, s_{k+1}$. As for each $i, 1 \leq i \leq k+1, s_i \subseteq S_i$ and $\{S_1, \ldots, S_{k+1}\}$ is a family of pairwise disjoint sets, the intersection property of the class Σ_k is violated in α. □

6 $\{V\Sigma_k\}_{1 \leq k \leq n}$ 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 that k-set agreement is weaker than k-simultaneous 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-simultaneous consensus. Hence, a protocol that uses a k-set agreement protocol to solve k-simultaneous 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 6.1).

6.1 From $V\Sigma_k$ to Σ_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 6.1. *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 $t < \frac{k'n}{k'+1}$.*

Proof. The proof of the theorem is divided in two parts. We first establish that for $k \leq k'$ or when $t < \frac{k'n}{k'+1}$, failure detector $\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}[V\Sigma_k]$. For the case $k \leq k'$, a simple protocol emulating a failure $\Sigma_{k'}$ is presented (Figure 3). When $t < \frac{k'n}{k'+1}$, $\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}$ (Theorem 5.5), and thus also in $\mathcal{MP}_{n,t}[V\Sigma_k]$. We then prove that if $k > k'$ and $t \geq \frac{k'n}{k'+1}$, it is not possible to emulate $\Sigma_{k'}$ (Lemma 6.3). The proof is based on a simple partitioning argument. □

Emulation of $\Sigma_{k'}$ in $\mathcal{MP}_{n,n-1}[V\Sigma_k]$, $k' \geq k$ A protocol that emulates a failure detector $\Sigma_{k'}$ is described in Figure 3. The protocol tolerates any number of failures, i.e., $t \leq n-1$. Each process p_i maintains a variable QUORUM_i that contains the output of the emulated failure detector $\Sigma_{k'}$.

Recall that the underlying failure detector $V\Sigma_k$ provides at each process p_i an array $VQ_i[1..k]$ of sets of processes. By the intersection property of the class $V\Sigma_k$, the sets output at any time and at any process in the cth entry intersect, for any $c, 1 \leq c \leq k$. So, any collection of $k'+1$ sets $Q_1, \ldots, Q_{k'+1}$ output by failure detector $V\Sigma_k$ contains at least two sets with a non-empty intersections since $k' \geq k$. That is, if for each $Q_i, 1 \leq i \leq k'+1$ there exists a process p_i, a time τ_i and an entry c_i such that $VQ_i[c_i] = Q_i$ at time τ_i, then two sets Q_j, Q_ℓ, $1 \leq j \neq \ell \leq k'+1$ are output in the same entry $c = c_i = c_\ell$ and thus are not disjoint. Therefore, the intersection property of the class $\Sigma_{k'}$ is satisfied if for each process p_i, QUORUM_i always contains some set output by $V\Sigma_k$. This is what is done in the protocol of Figure 3: periodically, the underlying failure detector is queried, and one of the sets output by $V\Sigma_k$ is selected as the new output of $\Sigma_{k'}$ (Task T1, lines 3–5).

It is also required that the output of the emulated failure detector eventually always contains at each correct process a set of correct processes. By the liveness property of the class $V\Sigma_k$, for each correct process p_i, there exist an entry $\ell_i, 1 \leq \ell_i \leq k$ such that $VQ_i[\ell_i]$ eventually always contains a set of correct processes. In order to eventually select one of the sets of correct processes output by $V\Sigma_k$, each process periodically broadcasts HEARTBEAT messages (task T1). As in the extraction protocol (Figure 1), HEARTBEAT are used to rank processes id. Each 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 T2). As each faulty process sends finitely many HEARTBEAT messages, there exists a time after which each correct process id appears before any faulty process id. Therefore, given two sets S, S' of process, $S \subseteq Correct$ and $S' \not\subseteq Correct$, 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 QUORUM_i, it selects a set S that has the smallest largest rank of its ids among the sets currently output by $V\Sigma_k$ (line 4). Since the sets output by $V\Sigma_k$ eventually contain a subset of the correct processes, this guarantees that eventually $\text{QUORUM}_i \subseteq Correct$. A proof of the protocol follows.

Lemma 6.2. *For all $k, k' : 1 \leq k \leq k' \leq n$ and $t : 1 \leq t \leq n-1$, the protocol described in Figure 3 implements a failure detector of the class $\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[V\Sigma_k]$.*

Proof. We show that the values of the variables QUORUM_i, for each process p_i, fulfill the specification of the class $\Sigma_{k'}$.

- *Intersection.* Let us consider $k'+1$ set $Q_1, \ldots, Q_{k'+1}$ output by the emulated failure detector. We have to show that at least two of them have a non-empty intersection.

  $Q_1, \ldots, Q_{k'+1}$ are the values of the variables $\text{QUORUM}_{i_1}, \ldots, \text{QUORUM}_{i_{k'+1}}$ at some times $\tau_1, \ldots, \tau_{k'+1}$ respectively for some (not necessarily distinct) processes $p_{i_1}, \ldots, p_{i_{k'+1}}$. By the code of the protocol (line 4), each Q_j is output by the underlying failure detector $V\Sigma_k$. More precisely, for each Q_j, there exists an integer $c_j, 1 \leq c_j \leq k$ and a time at

```
init   order_i ← (1, ..., n);                          /* ordered list of processes ids */
       QUORUM_i ← Π;                                    /* Σ_k emulated output */
       start task T1,T2,T3

task T1:
(1)    repeat periodically
(2)      send HEARTBEAT(i) to all processes;
(3)      VQ_i ← VΣ_k-query();        /* query VΣ_k; VQ_i is an array of k sets of procs. ids */
(4)      let Q ⊆ Π : ∃j : Q = VQ_i[j] and
                     ∀c, 1 ≤ c ≤ k : max_{ℓ∈Q} rank(ℓ, order_i) ≤ max_{ℓ∈VQ_i[c]} rank(ℓ, order_i)
                                  /* rank(j, order) is the rank of j in the ordered list order_i */
(5)      QUORUM_i ← Q
(6)    end repeat

task T2: when HEARTBEAT(j) is received:
(7)    move j at the head of the list order_i

task T3: when Σ_{k'} is queried:
(8)    return QUORUM_i
```

Figure 3: Emulation of $\Sigma_{k'}$ in $\mathcal{MP}_{n,n-1}[V\Sigma_k]$, $k' \geq k$ (code for p_i)

which at process p_{i_j}, the c_ith entry of the output of $V\Sigma_k$ is Q_i. As $k' \geq k$, for at least two sets Q_j, Q_ℓ, $\ell \neq j$, $c_j = c_\ell$ by the pigeonhole principle. Therefore, it follows from the intersection property of $V\Sigma_k$ that $Q_j \cap Q_\ell \neq \emptyset$.

- *Liveness* Let p_i denote a correct process. Periodically, the value of the variable QUORUM_i is refreshed with one of the sets currently output by the underlying failure detector $V\Sigma_k$. The set is chosen according to the current order on the processes ids given by the ordered list $order_i$ (lines 3–4).

  Let S, S' denote two sets of processes ids. The set S has higher rank than S' if every id in S appears before at least one id in S' in the list $order_i$. Suppose that $S \subseteq Correct$ whereas S' contains the id of a faulty process. A faulty process is moved finitely many times at the head of the list (line 7) since it sends finitely many HEARTBEAT messages. On the contrary, a correct process is moved infinitely many times at the beginning of the list (line 7). Hence, S is eventually always ranked before S'.

  By the liveness property of the class $V\Sigma_k$, there is a time after which the vector of sets output by the failure detector at p_i always contains a set of correct processes ids. It thus follows that eventually, a set of correct processes ids is selected each time line 4 is executed, from which we conclude that QUORUM_i eventually contains a set of correct processes ids. □

Impossibility of t-resilient emulation of $\Sigma_{k'}$ for $k < k'$ and $\frac{k'n}{k+1} \leq t$ To complete the proof of Theorem 6.1, we establish the following impossibility result:

Lemma 6.3. *Let n, t, k, k', $1 \leq k' < k < n$ and $\frac{k'n}{k'+1} \leq t$. There is no protocol that emulates a failure detector $\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[V\Sigma_k]$.*

Proof. The proof is based on a partition argument. Assume for contradiction that there exists a protocol $\mathcal{A}$ that emulates a failure detector $\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[V\Sigma_k]$.

As $\frac{k'n}{k'+1} \leq t$, i.e., $(k'+1)(n-t) \leq n$, there exists $k'+1$ pairwise disjoint sets $S_1, \ldots, S_{k'+1} \subseteq \Pi$ of size $n-t$. Let VQ be the vector of size k $[S_1, \ldots, S_{k'+1}, \Pi, \ldots, \Pi]$. Note that VQ is a valid

output for a failure detector of the class $V\Sigma_k$ in any execution in which the set of correct processes is Π or S_i, for any $i, 1 \leq i \leq k'+1$.

Let $\alpha_1, \ldots, \alpha_{k'+1}$ denote $k'+1$ executions of $\mathcal{A}$ as follows. For each $i, 1 \leq i \leq k'+1$, the set of correct processes in execution α_i is S_i; the output of the underlying failure detector $V\Sigma_k$ is always VQ, for every process $p_\ell \in S_i$. Processes in $\Pi \setminus S_i$ fail before taking any step. Observe that $n-t$ processes are correct in S_i.

Consider execution α_i, for some $i, 1 \leq i \leq k'+1$. By the correctness of the emulation $\mathcal{A}$, there exists a time τ_i at which the output of $\Sigma_{k'}$ is a set $s_i \subseteq S_i$, for some process in S_i. This follows from the liveness property of the class $\Sigma_{k'}$.

To establish a contradiction, we construct an execution α of $\mathcal{A}$ by "merging" execution $\alpha_1, \ldots, \alpha_{k'+1}$. Let $\tau = \max_{1 \leq i \leq k'+1} \tau_i$. No processes fail in α. As in each execution α_i, the output of the underlying failure detector $V\Sigma_k$ is always $VQ = [S_1, \ldots, S_{k'+1}, \Pi, \ldots, \Pi]$ as in $\alpha_i, 1 \leq i \leq k$. This is a valid output for a failure detector of the class $V\Sigma_k$, as no processes fail in α. Moreover, for each $i, 1 \leq i \leq k$, each message sent by the processes in S_i before time τ is delayed if it is sent to a process outside S_i, or received as in α_i. After time τ, we let every delayed messages be received in any order. Also, each message sent after time τ is eventually received.

It thus follows that run α and α_i are indistinguishable up to time τ by each process in S_i. Therefore, by construction of α_i, at time τ_i and at some process S_i, the output of $\mathcal{A}$ is a quorum $s_i \subseteq S_i$. As $S_1, \ldots, S_{k'+1}$ is a family of pairwise disjoint sets, so is the family $s_1, \ldots, s_{k'+1}$. Therefore, the intersection property of $\Sigma_{k'}$ is violated: a contradiction. □

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

The next theorem complements Theorem 6.1 by characterizing the values of the parameters t, n and k for which it is possible to emulate $V\Sigma_k$ when a failure detector Σ_k is available.

Theorem 6.4. *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 5.2 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 $\mathcal{MP}_{n,t}[\Sigma_k]$. We consider two cases according to the value of t:

- $t < \frac{k'n}{k'+1}$. In that case, we know from Theorem 5.5 that Σ_k can be emulated in $\mathcal{MP}_{n,t}$. Therefore, by combining this emulation with $\mathcal{A}$, it follows that $V\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}$. As $t \geq \frac{n+k'-2}{2}$, this contradicts Theorem 5.1.

- $t \geq \frac{k'n}{k'+1}$. In that case, $n \geq (k'+1)(n-t)$ and thus there are $k'+1$ pairwise disjoint sets $S_1, \ldots, S_{k'+1} \subseteq \Pi$ of size $n-t$.

  For each $i, 1 \leq i \leq k'+1$, 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 $s_i \subseteq S_i$.

  As for each $i, 1 \leq c_i \leq k'$, there exists $j, \ell, 1 \leq j \neq \ell \leq k'+1$ such that $c_j = c_\ell = c$. We obtain a contradiction by merging executions α_i and α_j. Specifically, 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 6.5. *Let $n, t, k, k', 1 \leq k, k', t < n$. There is no protocol that emulates a failure detector $V\Sigma_{k'}$ in $\mathcal{MP}_{n,t}[\Sigma_k]$ if $k \geq 2$ and $t > \frac{n+k'-2}{2}$.*

Proof. As in the previous proof, we consider two cases according to the value of t.

- $t < \frac{k'n}{k'+1}$. In this case Σ_k can be emulated in $\mathcal{MP}_{n,t}$ (Theorem 5.5), and consequently, $V\Sigma_{k'}$ can be emulated in $\mathcal{MP}_{n,t}[\Sigma_k \times \Omega]$ if and only if it can be emulated in $\mathcal{MP}_{n,t}[\Omega]$. By Lemma 5.4, $V\Sigma_{k'}$ cannot be emulated in $\mathcal{MP}_{n,t}[\Omega]$ if $t > \frac{n+k'-2}{2}$.

- $t \geq \frac{k'n}{k'+1}$. Essentially the same strategy as in the previous proof can be reused. What is left undefined in the execution considered there is the output of failure detector Ω. In each $\alpha_i, 1 \leq i \leq k'+1$, Ω may always output the same process $\in S_i$ 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. □

7 Separation results

This section glues together the results presented in Section 4, Section 5 and Section 6 to establish the following main theorem:

Theorem 7.1. *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'-simultaneous consensus in $\mathcal{MP}_{n,t}[k\text{-}SA]$.*

The proof strategy is as follows: On one hand, it has been shown [11] 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'-simultaneous consensus (Theorem 4.1. The impossibility of a t-resilient solution to k'-simultaneous consensus using k-set agreement protocols 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 6.5.

Proof. Let $k \geq 2$ and $t > \frac{n+k'-2}{2}$. The proof is by contradiction. Assume that there exists a k'-simultaneous 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'-simultaneous 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 $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$ [11] – the protocol presented there imposes no requirement on t and k. $\mathcal{B}$ may thus be the protocol presented in [11]. Therefore, by combining protocol $\mathcal{A}$ and $\mathcal{B}$, it follows that k'-simultaneous consensus can be solved in $\mathcal{MP}_{n,t}[\Omega \times \Sigma_k]$.

In section 4, we have shown that $V\Sigma_k$ is necessary for k'-simultaneous consensus. That is, if there is a k'-simultaneous 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'-simultaneous 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 6.5, 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-simultaneous consensus is also expressed by the following theorem. The theorem gives tight bounds on t for k-set agreement and k-simultaneous consensus to be solvable when an eventual leader is available:

Theorem 7.2. *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-simultaneous consensus protocol if and only if $t \leq \frac{n+k-2}{2}$.*

Proof. 1. Bonnet and Raynal[21] 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 [11]. 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-simultaneous (Section 4) 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 5.4). Therefore, no protocol solves k-simultaneous consensus in $\mathcal{MP}_{n,t}[\Omega]$ if $t > \frac{n+k-2}{2}$.

   For $t \leq \frac{n+k-2}{2}$, k-simultaneous consensus can be solved by having each process p_i participate simultaneously in k instances $\mathcal{A}^1, \ldots, \mathcal{A}^k$ of an $(\Omega \times \Sigma)$-based consensus protocol $\mathcal{A}$. Process p_i proposes its initial value in each instance; the first time a value v is decided by p_i in an instance of consensus, the pair (c, v) is returned by p_i as its output for k-simultaneous consensus, where c is the instance of consensus in which it decides. Emulation of Σ in instance j consists in outputting the jth entry 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 $v = v'$. □

References

[1] Yehuda Afek, Eli Gafni, Sergio Rajsbaum, Michel Raynal, and Corentin Travers. The k-simultaneous consensus problem. *Distributed Computing*, 22(3):185–195, 2010.

[2] Marcos K. Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. Partial synchrony based on set timeliness. *Distributed Computing*, 25(3):249–260, 2012.

[3] Dan Alistarh, Seth Gilbert, Rachid Guerraoui, and Corentin Travers. Of choices, failures and asynchrony: The many faces of set agreement. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC '09, pages 943–953, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. *J. ACM*, 42(1):124–142, 1995.

[5] J. Bárány. A short proof of kneser's conjecture. *Journal of Combinatorial Theory, Series A*, 25(3):325–326, 1978.

[6] François Bonnet and Michel Raynal. A simple proof of the necessity of the failure detector sigma to implement an atomic register in asynchronous message-passing systems. *Inf. Process. Lett.*, 110(4):153–157, 2010.

[7] François Bonnet and Michel Raynal. On the road to the weakest failure detector for k-set agreement in message-passing systems. *Theor. Comput. Sci.*, 412(33):4273–4284, 2011.

[8] Elizabeth Borowsky and Eli Gafni. Generalized flp impossibility result for t-resilient asynchronous computations. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 91–100. ACM, 1993.

[9] Zohir Bouzid and Corentin Travers. (anti-omega$^x \times \sigma_z$-based k-set agreement algoritms. In *Proceedings 14th International Conference on Principles of Distributed Systems (OPODIS)*, volume 6490 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2010.

[10] Tushar Chandra, Vassos Hadzilacos, and Sam Toueg. The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722, 1996.

[11] Tushar Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[12] Soma Chaudhuri. More choices allow more faults: set consensus problems in totally asynchronous systems. *Inf. Comput.*, 105(1):132–158, 1993.

[13] Carole Delporte-Gallet, Hugues Fauconnier, and Rachid Guerraoui. Tight failure detection bounds on atomic object implementations. *J. ACM*, 57(4), 2010.

[14] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Tielmann. The weakest failure detector for message passing set-agreement. In *Proceedings 22nd International Symposium on Distributed Computing (DISC)*, volume 5218 of *Lecture Notes in Computer Science*. Springer, 2008.

[15] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Tielmann. The disagreement power of an adversary. *Distributed Computing*, 24(3-4):137–147, 2011.

[16] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[17] Eli Gafni and Rachid Guerraoui. Generalized universality. In *Proceedings 22nd International Conference on Concurrency Theory (CONCUR)*, volume 6901 of *Lecture Notes in Computer Science*, pages 17–27. Springer, 2011.

[18] Eli Gafni and Petr Kuznetsov. The weakest failure detector for solving k-set agreement. In *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 83–91. ACM, 2009.

[19] Eli Gafni, Sergio Rajsbaum, Michel Raynal, and Corentin Travers. The committee decision problem. In *Proceedings 7th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 3887 of *Lecture Notes in Computer Science*, pages 502–514. Springer, 2006.

[20] J.E. Greene. A new short proof of kneser's conjecture. *The American mathematical monthly*, 109(10):918–920, 2002.

[21] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.

[22] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[23] L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

[24] J. Matoušek. A combinatorial proof of knesers conjecture*. *Combinatorica*, 24(1):163–170, 2004.

[25] Achour Mostéfaoui, Sergio Rajsbaum, Michel Raynal, and Corentin Travers. On the computability power and the robustness of set agreement-oriented failure detector classes. *Distributed Computing*, 21(3):201–222, 2008.

[26] Achour Mostéfaoui, Michel Raynal, and Julien Stainer. Relations linking failure detectors associated with k-set agreement in message-passing systems. In *Proceedings 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 6976 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 2011.

[27] Philippe Raipin Parvédy, Michel Raynal, and Corentin Travers. Strongly terminating early-stopping k-set agreement in synchronous systems with general omission failures. *Theory Comput. Syst.*, 47(1):259–287, 2010.

[28] Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.

[29] Piotr Zielinski. Anti-*omega*: the weakest failure detector for set agreement. *Distributed Computing*, 22(5-6):335–348, 2010.