Weakening Failure Detectors for k-Set Agreement via the Partition Approach

Wei Chen[1], Jialin Zhang[2]*, Yu Chen[1], Xuezheng Liu[1]

[1] Microsoft Research Asia
{weic, ychen, xueliu}@microsoft.com

[2] Center for Advanced Study
Tsinghua University
zhanggl02@mails.tsinghua.edu.cn

Abstract. In this paper, we propose the partition approach and define several new classes of partitioned failure detectors weaker than existing failure detectors for the k-set agreement problem in both the shared-memory model and the message-passing model. In the shared-memory model with $n+1$ processes, for any $2 \leq k \leq n$, we first propose a partitioned failure detector $\Pi\Omega_k$ that solves k-set agreement with shared read/write registers and is strictly weaker than Ω_k, which was conjectured to be the weakest failure detector for k-set agreement in the shared-memory model [19]. We then propose a series of partitioned failure detectors that can solve n-set agreement, yet they are strictly weaker than Υ [10], the weakest failure detector ever found before our work to circumvent any asynchronous impossible problems in the shared-memory model. We also define two new families of partitioned failure detectors in the message-passing model that are strictly weaker than the existing ones for k-set agreement. Our results demonstrate that the partition approach opens a new dimension for weakening failure detectors related to set agreement, and it is an effective approach to check whether a failure detector is the weakest one or not for set agreement. So far, all previous candidates for the weakest failure detectors of set agreement have been disproved by the partitioned failure detectors.

Keywords: failure detector, partitioned failure detectors, k-set agreement.

1 Introduction

Failure detector abstractions are first proposed by Chandra and Toueg in [3] to circumvent the impossibility result of consensus [9], and have since become a powerful technique to encapsulate system conditions needed to solve many distributed computing problems. Among them the problem of k-set agreement has received many attention from the research community. Informally, in k-set agreement each process proposes

* This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

some value and eventually all correct processes (those that do not crash) decide on at most k different values [4]. It has been shown that k-set agreement cannot be solved in asynchronous systems when k or more processes may crash [1, 12, 20]. In recent years, a number of studies have focused on failure detectors for solving k-set agreement problem [21, 18, 11, 16, 17, 19, 10, 7]. These studies form the collective effort in the pursuit of the weakest failure detector for k-set agreement, a goal yet to be reached. A particular candidate Ω_k was conjectured to be the weakest failure detector for wait-free k-set agreement [19] in the shared-memory model.

Consider distributed shared-memory model with $n + 1$ processes. In a very recent paper [10], Guerraoui et.al define a new class of failure detectors Υ and show that among a wide range of failure detectors defined as *eventually stable failure detectors*, Υ is the weakest one necessary to solve *any* impossible problem in shared-memory distributed systems, and Υ solves the n-set agreement problem. The Υ failure detector disproves the conjecture on Ω_k for the case of $k = n$. For a general k, a generalized Υ^k is proposed to solve k-set agreement, but only when at most k processes may crash, so it does not disprove the conjecture on Ω_k for wait-free k-set agreement.

The eventually stable failure detectors encompass most failure detectors known to solve distributed decision tasks in the shared-memory model prior to [10], as the authors claimed. Therefore, as the title of their paper says, indeed Υ is the weakest failure detector ever found that solves any impossible problem in distributed computing.

In this paper, we introduce a new breed of failure detectors — *partitioned failure detectors* — that could be made strictly weaker than Ω_k and Υ but are still strong enough to solve the set agreement problem. Our motivation is based on the following observation: In k-set agreement when $k > 1$, different processes may decide on different values, and thus it is possible that processes may be partitioned to different components, each of which decides on different values but together they still decide on at most k values. In other words, k-set agreement (with $k > 1$) exhibits the partition nature. The partitioned failure detectors are defined by consistently applying a method that captures the partition nature to weaken existing failure detectors, for which we called the *partition approach*.

In the partition approach, failure detectors partition the processes into multiple components and only processes in one of the components (called a *live component*) are required to satisfy all safety and liveness properties (of an existing failure detector), while processes in other components only need to satisfy safety properties. Since those processes in non-live components may generate quite arbitrary failure detector outputs, intuitively the partitioned failure detectors are a new breed that does not fall into the eventually stable failure detectors covered by [10].

We study the partitioned failure detectors in both the shared-memory model and the message-passing model. In the main part of this paper, we apply the partition approach to failure detectors Ω_k and Υ in the shared-memory model to define weaker failure detectors. More specifically, we first define a new class of failure detectors $\Pi\Omega_k$ by applying static partitions to Ω_k. We show that $\Pi\Omega_k$ is strong enough to solve k-set agreement with shared read/write registers but it is not comparable with Υ, for all $k = 2, 3, \ldots, n$. One direct consequence is that $\Pi\Omega_k$ is strictly weaker than Ω_k (because Ω_k is stronger than Υ), which disproves the conjecture that Ω_k is the weakest failure

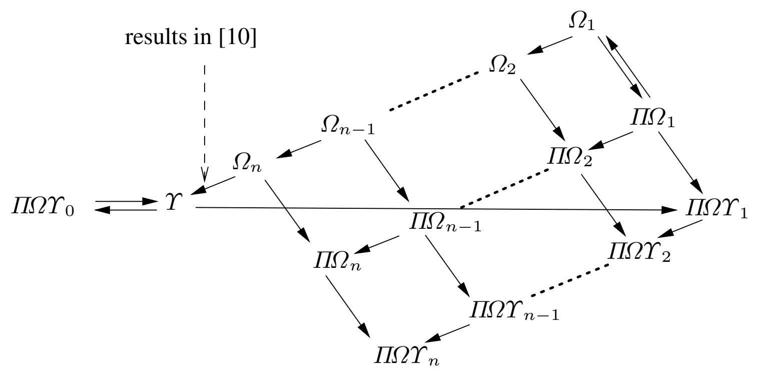

Fig. 1. Relationship diagram for failure detectors in the shared-memory model ($n \geq 3$). If $A \rightarrow B$, then A can be transformed into B. If there is no directed path from A to B, then A cannot be transformed into B (Footnote 1 contains the only exception).

detector for wait-free k-set agreement in the shared-memory model for any $k \geq 2$. Moreover, $\Pi\Omega_k$ is the first failure detector class that solves k-set agreement (for generic k) but is incomparable with Υ. For example, even though failure detector $\Pi\Omega_2$ solves 2-set agreement, it is not stronger than Υ.

Next, we define failure detectors weaker than Υ but are still strong enough to solve n-set agreement. We achieve this by mixing some of the properties of $\Pi\Omega_k$ and Υ and define another class of partitioned failure detectors $\Pi\Omega\Upsilon_k$. We show that for any $1 \leq k \leq n$, $\Pi\Omega\Upsilon_k$ can still solve n-set agreement but it is strictly weaker than both $\Pi\Omega_k$ and Υ. Moreover, as k increases, the strength of $\Pi\Omega\Upsilon_k$ is strictly weakened. Hence, we find a family of n different failure detector classes strictly weaker than Υ, which is the weakest one ever found before our work.

Figure 1 characterizes the exact relationship among all failure detectors we proposed in this paper for the shared-memory model and the previously defined ones Ω_k and Υ. Note that every nonexistent directed path in the figure corresponds to an impossible transformation from the source class to the destination class, with only one exception.[1] Since Υ is already very weak, one can imagine that it would be very delicate to define the new partitioned failure detectors and prove that they are incomparable to or strictly weaker than Υ. Indeed, the definitions of failure detectors are subtle, and the proofs of the impossible transformations are the most delicate and technically involved.

We also apply the partition approach to failure detectors $\Omega_k \times \Sigma$ in the message-passing model, where Σ is the class of quorum failure detectors needed to work with Ω_k to solve k-set agreement in the message-passing model. We define two new families of partitioned failure detectors that are strictly weaker than $\Omega_k \times \Sigma$ but are strong enough to solve k-set agreement in the message-passing model. These partitioned failure detectors are different from the ones in the shared-memory model in that they integrate

[1] The exception is the following problem that is still open: Can $\Pi\Omega_k$ be transformed into $\Pi\Omega\Upsilon_{k-1}$ for any $k \geq 2$? However, we have proven that $\Pi\Omega_{k+1}$ cannot be transformed into $\Pi\Omega\Upsilon_{k-1}$ for any $k \geq 2$.

the partition of quorums in their definitions. Moreover, one family of failure detectors incorporates dynamic splitting of partitions, while all failure detectors in the shared-memory model are statically partitioned.

Our results not only show a number of new failure detectors that are strictly weaker than existing ones such as $\varOmega_k$ and $\varUpsilon$, but more importantly, they demonstrate the power of the partition approach: The partition approach opens a new dimension for weakening various failure detectors related to set agreement, and it is an effective approach to check whether a failure detector could be the weakest one solving set agreement or not. Using the approach, we have successfully shown that (1) $\varOmega_k$ is not the weakest failure detector for k-set agreement in the shared-memory model for any $k \geq 2$; (2) $\varUpsilon$ is not the weakest failure detector for n-set agreement in the shared-memory model; and (3) $\varOmega_k \times \varSigma$ is not the weakest failure detector for k-set agreement in the message-passing model for any $k \geq 2$. So far, all failure detectors that were considered as the candidates for the weakest failure detectors for set agreement have been disproved using our partition approach. Therefore, we believe that partitioned failure detectors demonstrate the flexibility in achieving set agreement, and it is important to use the partition approach as an effective research tool in our pursuit to the ultimate weakest failure detectors for set agreement.

The rest of the paper is organized as follows. Section 2 provides the shared-memory model used in our paper. Section 3 defines $\varPi\varOmega_k$ and shows how it solves k-set agreement. Section 4 defines $\varPi\varOmega\varUpsilon_k$. Section 5 provides a central place to show the relationship among all failure detectors in the shared-memory model as captured by Figure 1. Section 6 summarizes the results in the message-passing model. We conclude the paper in Section 7. Further results including some k-set agreement algorithms and all correctness proofs are covered by two technical reports [7, 5] on message-passing model and shared-memory model, respectively.

2 Model

We consider asynchronous shared-memory distributed systems augmented with failure detectors. Our model is the same as the model in [10], which is based on the models of [13, 14, 2]. We provide the necessary details of the model below.

We consider a system with $n+1$ processes $P = \{p_1, p_2, \ldots, p_{n+1}\}$ where $n \geq 1$. Let $\mathcal{T}$ be the set of global time values, which are non-negative integers. Processes do not have access to the global time. A *failure pattern* F is a function from $\mathcal{T}$ to 2^P, such that $F(t)$ is the set of processes that have failed by time t. Failed processes do not recover, i.e., $F(t) \subseteq F(t+1)$ for all $t \in \mathcal{T}$. Let *correct*(F) denote the set of *correct processes*, those that do not crash in F. A process is *faulty* if it is not correct. A *failure detector history* H is a function from $P \times \mathcal{T}$ to an output range $\mathcal{R}$, such that $H(p, t)$ is the output of the failure detector module of process $p \in P$ at time $t \in \mathcal{T}$. A *failure detector* $\mathcal{D}$ is a function from each failure pattern to a set of failure detector histories, representing the possible failure detector outputs under failure pattern F.

Processes communicate with each other by writing to and reading from shared atomic registers. A deterministic algorithm A using a failure detector $\mathcal{D}$ is a collection of $n+1$ deterministic automata, one for each process. Processes execute by taking *steps*. In each step, a process p: (a) reads from a shared register to obtain a value, or writes a value to a shared register, or queries its failure detector module, based on its current local state; and (b) transitions its current state to a new state, based on its current state, the value returned from the read or from the failure detector module, and the algorithm automaton on p. Each step is completed at one time point t, but the process may crash in the middle of taking its step. A *run* of algorithm A with failure detector $\mathcal{D}$ under a failure pattern F is an infinite sequence of steps such that every correct process takes an infinite number of steps and no faulty process takes any step after it crashes.

We say that a failure detector class $\mathcal{C}_1$ is *weaker than* a failure detector class $\mathcal{C}_2$, if there is a transformation algorithm T such that using any failure detector $\mathcal{D}_2 \in \mathcal{C}_2$, algorithm T implements a failure detector $\mathcal{D}_1 \in \mathcal{C}_1$. By implementing $\mathcal{D}_1$ we mean that for any run of algorithm T with failure detector $\mathcal{D}_2$ under a failure pattern F, T generates the outputs of $\mathcal{D}_1$ as a distributed variable $\mathcal{D}_1$-*output* such that there exists failure detector history $H \in \mathcal{D}_1(F)$ and $H(p,t) = \mathcal{D}_1$-*output*(p,t) for all $p \in P$ and all $t \in \mathcal{T}$, where $\mathcal{D}_1$-*output*(p,t) is the value of the variable $\mathcal{D}_1$-*output* on p at time t. If $\mathcal{C}_1$ is weaker than $\mathcal{C}_2$, we denote it as $\mathcal{C}_1 \preceq \mathcal{C}_2$ and also refer to it as $\mathcal{C}_2$ can be transformed into $\mathcal{C}_1$. if $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \not\preceq \mathcal{C}_1$, we say that $\mathcal{C}_1$ is *strictly weaker than* $\mathcal{C}_2$ and denote it as $\mathcal{C}_1 \prec \mathcal{C}_2$. If $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \preceq \mathcal{C}_1$, we say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent and denote it as $\mathcal{C}_1 \equiv \mathcal{C}_2$.

In k-set agreement with $1 \leq k \leq n$, each process proposes a value, and makes an irrevocable decision on one value. It needs to satisfy the following three properties: (1) *Validity*: If a process decides v, then v has been proposed by some process. (2) *Uniform k-Agreement*: There are at most k different decision values. (3) *Termination*: Eventually all correct processes decide.

Two related failure detector classes are Ω_k and Υ. Failure detectors in Ω_k output a subset of P of size at most k, and there is a time after which all processes always output the same nonempty set, which contains at least one correct processes. Failure detectors in Υ also output a subset of P, and there is a time after which all processes always output the same nonempty set, which is not exactly the set of correct processes.

3 Failure Detector $\Pi\Omega_k$

3.1 Specification of $\Pi\Omega_k$

The class of partitioned failure detectors $\Pi\Omega_k$ is obtained by applying static partitions to Ω_k, as explained below. The output of $\Pi\Omega_k$ for process p is a tuple (*isLeader*, *lbound*, *cid*), where *isLeader* is a boolean value indicating whether this process is a leader or not, *lbound* is a non-negative integer indicating the upper bound on the number of possible leaders in p's partitioned component, and *cid* is a component ID drawn from an ID set $\mathcal{I}$ or is a special value $\perp \notin \mathcal{I}$. The *cid* output indicates the component the process belongs to and could be $\perp$ for an initial period before the failure detector decides on a partition.

For a failure detector output x, we use $x.v$ to denote the field v of x, where v could be *isLeader*, *lbound*, or *cid* in the case of $\Pi\Omega_k$. We say that a process p is an *eventual leader* (under a failure pattern F and a failure detector history H) if p is correct and there is a time after which the *isLeader* output on p is always *True*.

A *partition* of P is $\pi = \{P_1, \ldots, P_s\}$, where $s \geq 1$ and P_i's are non-empty subsets of P such that they do not intersect with one another and their union is P. For a process p, we use $\pi[p]$ to denote the partitioned component that contains p. For a component $P_j \subseteq P$ (under a failure pattern F and a failure detector history H), we define $\mathit{lbound}(P_j) = \max\{H(p,t).\mathit{lbound} \mid t \in \mathcal{T}, p \in P_j \setminus F(t)\}$,[2] and $\mathit{Leaders}(P_j) = \{p \in P_j \cap \mathit{correct}(F) \mid \exists t, \forall t' > t, H(p,t').\mathit{isLeader} = \mathit{True}\}$. The value $\mathit{lbound}(P_j)$ is the maximum *lbound* value among processes in component P_j, while $\mathit{Leaders}(P_j)$ is the set of eventual leaders in P_j.

A failure detector $\mathcal{D}$ is in the class $\Pi\Omega_k$ if for any failure pattern F and any failure detector history $H \in \mathcal{D}(F)$, there exists a partition $\pi = \{P_1, \ldots, P_s\}$ of P, such that the following properties hold. First, the *cid* output needs to satisfy these properties:

($\Pi C1$) The *cid* outputs on all correct processes eventually always output non-$\perp$ values. Formally, $\exists t_0 \in \mathcal{T}, \forall p \in \mathit{correct}(F), \forall t \geq t_0, H(p,t).\mathit{cid} \neq \perp$.

($\Pi C2$) The non-$\perp$ *cid* outputs distinguish different components. Formally, $\forall t_1, t_2 \in \mathcal{T}, \forall p_1 \notin F(t_1), \forall p_2 \notin F(t_2), (H(p_1,t_1).\mathit{cid} \neq \perp \wedge H(p_2,t_2).\mathit{cid} \neq \perp) \Rightarrow ((H(p_1,t_1).\mathit{cid} = H(p_2,t_2).\mathit{cid}) \Leftrightarrow (\pi[p_1] = \pi[p_2]))$.

Next, the *isLeader* and *lbound* outputs satisfy the following set of safety and liveness properties. The safety property is:

($\Pi\Omega1$) The sum of the maximum *lbound* outputs in all partitioned components does not exceed k. Formally, $\sum_{j=1}^{s} \mathit{lbound}(P_j) \leq k$.

The liveness part specifies that there exists one partitioned component P_j such that:

($\Pi\Omega2$) Eventually *lbound* outputs by all processes in P_j are the same. Formally, $\exists t_0 \in \mathcal{T}, \forall t_1, t_2 \geq t_0, \forall p_1 \in P_j \setminus F(t_1), \forall p_2 \in P_j \setminus F(t_2), H(p_1,t_1).\mathit{lbound} = H(p_2,t_2).\mathit{lbound}$.

($\Pi\Omega3$) Eventually the *isLeader* outputs on any correct process in P_j do not change. Formally, $\exists t_0 \in \mathcal{T}, \forall t > t_0, \forall p \in P_j \setminus F(t), H(p,t).\mathit{isLeader} = H(p,t_0).\mathit{isLeader}$.

($\Pi\Omega4$) There is at least one eventual leader. Formally, $|\mathit{Leaders}(P_j)| \geq 1$.

($\Pi\Omega5$) The number of eventual leaders is eventually bounded by the *lbound* outputs. Formally, $\exists t_0 \in \mathcal{T}, \forall t \geq t_0, |\mathit{Leaders}(P_j)| \leq H(p,t).\mathit{lbound}$.

We call a component that satisfies the liveness properties ($\Pi\Omega2$–5) a *live component*, and other components *non-live components*. Let $k_i = \mathit{lbound}(P_i)$. Intuitively, each component P_i has a failure detector with the safety properties of Ω_{k_i} restricted to P_i,[3] while at least one component P_j also satisfies all liveness properties of Ω_{k_j}. Intuitively, this is to guarantee that when running a k-set agreement algorithm with $\Pi\Omega_k$, each component P_i may decide on at most k_i values, so with ($\Pi\Omega1$) there are at most k decisions, while the live component P_j can make progress and decide eventually.

[2] As a convention, $\max \emptyset = 0$.

[3] In [6] we show that a variation of failure detectors that output *isLeader* and *lbound*, named Ω''_k, is equivalent to Ω_k failure detectors.

The strength of $\Pi\Omega_k$ is fully characterized by Figure 1. We defer to Section 5 as a central place to study and compare the strength of all proposed failure detectors and avoid repetitions. We summarize the strength of $\Pi\Omega_k$ comparing with Ω_k and Υ in the following theorem.

Theorem 1. *The followings hold regarding the strength of $\Pi\Omega_k$. (1) $\Pi\Omega_1 \equiv \Omega_1$. (2) $\Pi\Omega_k \prec \Omega_j$ for all $k \geq 2$, $j \geq 1$, and $k \geq j$. (3) $\Pi\Omega_k \not\preceq \Omega_j$ and $\Omega_j \not\preceq \Pi\Omega_k$ for all $k \geq 2$ and $k < j \leq n$. (4) $\Pi\Omega_k \prec \Pi\Omega_{k-1}$ for all $k \geq 2$. (5) $\Pi\Omega_k \not\preceq \Upsilon$ and $\Upsilon \not\preceq \Pi\Omega_k$, for all $k \geq 2$.*

The key result is that $\Pi\Omega_k$ is incomparable with Υ for all $k \geq 2$. Therefore, $\Pi\Omega_k$ is a new class of failure detectors that is strictly weaker than Ω_k, but is strong enough to solve k-set agreement in shared-memory systems with arbitrary failure patterns. It is the only class known (to our best knowledge) that solves k-set agreement with arbitrary failure patterns and is strictly weaker than Ω_k and is incomparable with Υ.[4]

3.2 Solving k-set agreement with $\Pi\Omega_k$

The algorithm using $\Pi\Omega_k$ to solve k-set agreement is based on an extension of the k–$converge$ algorithm presented in [21]. The original k–$converge$ algorithm forces every participant to use the same value of "k". With $\Pi\Omega_k$ failure detectors, we need processes in each component to try to converge on some decisions, the number of which is bounded by the *lbound* output of the failure detector. Therefore we extend the k–$converge$ algorithm by moving "k" into the parameter of the routine and rename the routine to $converge()$. We adjust the specification of $converge()$ as follows.

Routine $converge()$ takes in three parameters: ℓ is the upper bound on the number of values can be committed (this parameter corresponds to the "k" in k–$converge$), p is the process identifier, and v is the input value of the process. It outputs a pair (c, v'), where c is a boolean and v' is one of the input value. When p outputs (c, v'), we say that p picks v', and if $c =$ *True*, we say that p commits to v'. The routine satisfies the following properties: (1) C-Termination: Every correct process picks some value. (2) C-Validity: If a process p picks value v, then some process q invoked $converge()$ with parameter v. (3) C-Agreement: If a process p commits to a value, then at most ℓ_{max} values are picked, where ℓ_{max} is the maximum ℓ that processes pass into $converge()$. (4) Convergence: If all processes use the same value in the ℓ parameter ($\ell > 0$), and if there are no more than ℓ distinct input values, then every process that picks a value commits. The first two properties are the same as in [21], while the last two properties are adjusted to accommodate different input values of ℓ. Although the interface and the specification are changed, the algorithm is exactly the same as in [21], and the proof only needs some minor adjustment. The algorithm and its proof are included in [5].

Based on the $converge()$ routine, we provide an algorithm to solve k-set agreement using $\Pi\Omega_k$ in Figure 2. The algorithm is straightforward. We use *cid* output of failure detectors to isolate each component and make sure only processes in the same component could run the same instance of $converge()$ routine. Within a component, only

[4] The Υ^k failure detector proposed in [10] only solves k-set agreement in systems with at most k failures.

```
Shared variables:
    Register $D$, initially $\bot$
    $converge()$ instances: $converge[\,][\,]$

Output of failure detector $\Pi\Omega_k$ on process $p_i$:
    $isLeader_i$, $lbound_i$, $cid_i$

Code for process $p_i$:
1   $v \leftarrow$ the input value of $p_i$
2   repeat
3       $cid \leftarrow cid_i$
4   until $cid \neq \bot$
5   $r \leftarrow 0$
6   repeat
7       $c \leftarrow False$
8       if $isLeader_i = True$ then
9           $r \leftarrow r + 1$
10          $(c, v) \leftarrow converge[cid][r](lbound_i, i, v)$
11      if $c = True$ then
12          $D \leftarrow v$; return $(D)$
13  until $D \neq \bot$
14  return $(D)$
```

Fig. 2. k-set agreement algorithm using $\Pi\Omega_k$

those processes with *isLeader* output being *True* can run $converge()$ instances. Each $converge()$ instance only uses the output of the previous $converge()$ instance as the input, which is important to guarantee the safety of the algorithm. In any $converge()$ instance if some process p commits to a value v, then p writes v to a shared variable D and decides on v, and eventually all correct processes will see a non-$\bot$ D value and decide. The following theorem summarizes the correctness of the algorithm.

Theorem 2. *Algorithm in Figure 2 solves k-set agreement using failure detectors in $\Pi\Omega_k$, for any $k \geq 1$.*

Proof. It's obvious that k-set *Validity* holds.

For *Uniform k-Agreement*, we only need to consider decisions made in line 12, since decisions made in line 14 do not generate new decision values. Consider every component P_i. If some process decides in line 12, we consider the earliest such decision, say by a process $p \in P_i$. Process p decides v because it commits to v in an instance $converge[cid][r]()$. By the *C-Agreement* property of $converge()$, at most ℓ_{max} values can be picked in this $converge[cid][r]()$ instance, where ℓ_{max} is the maximum *lbound* values in the input of this instance. Since the algorithm guarantees for any $r' > r$, instances $converge[cid][r']()$ only uses the values picked in instance $converge[cid][r]()$, we know that there are at most ℓ_{max} values can be decided in line 12 by processes in component P_i. By definition, $\ell_{max} \leq lbound(P_i)$. Then, by property ($\Pi\Omega 1$), there are at most k values that can be decided. So Uniform k-Agreement holds.

For k-set *Termination*, first by property ($\Pi C2$) all correct processes eventually exit the loop in lines 2–4. In the live component P_j that satisfies ($\Pi\Omega 2$–5), eventually there is at least one correct process and at most ℓ processes in P_j invoking $converge()$, where ℓ is the eventually converged *lbound* output value. Moreover, all these processes invoke $converge()$ with the same first parameter value ℓ. Thus, the *C-Termination* and *Convergence* properties guarantee that all correct processes in P_j eventually commit to some value in some $converge()$ instance. Therefore, eventually D is written. Once D is written, all correct processes eventually decide. □

4 Failure Detector $\Pi\Omega\Upsilon_k$

After defining $\Pi\Omega_k$, our next step is to find a mixture of $\Pi\Omega_k$ and Υ such that the new failure detectors are weaker than both and are still strong enough to solve n-set agreement. Since we know that $\Pi\Omega_k$ and Υ are not comparable, it immediately means that the new failure detectors are strictly weaker than both $\Pi\Omega_k$ and Υ. This leads us to the discovery of failure detectors $\Pi\Omega\Upsilon_k$.

The output of $\Pi\Omega\Upsilon_k$ for process p is a tuple $(S, \mathit{lbound}, \mathit{cid})$, where S is a subset of P that informally matches the output of Υ, and *lbound* and *cid* outputs have the same value range and same informal meaning as the ones in $\Pi\Omega_k$. For a component P_j, let $\mathit{correct}(P_j) = \mathit{correct}(F) \cap P_j$, the set of correct processes in P_j (under a failure pattern F).

A failure detector $\mathcal{D}$ is in the class $\Pi\Omega\Upsilon_k$ if for any failure pattern F and any failure detector history $H \in \mathcal{D}(F)$, there exists a partition $\pi = \{P_1, \ldots, P_s\}$ of P, such that the following properties hold. The *cid* properties and safety properties are the same as $\Pi\Omega_k$, namely $(\Pi C1)$, $(\Pi C2)$, and $(\Pi\Omega 1)$. The liveness part specifies that there exists one partitioned component P_j such that $(\Pi\Omega 2)$ of $\Pi\Omega_k$ and the following property hold:

$(\Pi\Upsilon 1)$ P_j contains at least one correct process, and eventually all correct processes in P_j output the same $S \subseteq P_j$ such that S is not the set of correct processes in P_j and either $S \neq \emptyset$ or the number of correct processes is bounded by the eventual *lbound* output. Formally, $\mathit{correct}(P_j) \neq \emptyset \wedge \exists S_0 \subseteq P_j, S_0 \neq \mathit{correct}(P_j), \exists t_0, (\forall p \in \mathit{correct}(P_j), \forall t > t_0, (H(p,t).S = S_0 \wedge (S_0 \neq \emptyset \vee |\mathit{correct}(P_j)| \leq H(p,t).\mathit{lbound})))$.

We call a component that satisfies the liveness properties $(\Pi\Omega 2)$ and $(\Pi\Upsilon 1)$ a *live component*, and other components *non-live components*. Intuitively, in the live component P_j, the S output behaves almost the same as the output of Υ, except that S may eventually stabilize to $\emptyset$, in which case the number of correct processes in P_j must be bounded by the eventual *lbound* output. This mixture is important in making $\Pi\Omega\Upsilon_k$ strictly weaker than Υ. In particular, $\Pi\Omega\Upsilon_0$ is well-defined since *lbound* outputs could always be 0. However, in $\Pi\Omega\Upsilon_0$ the above mixture of requirements on S and on *lbound* is gone, and we will show that $\Pi\Omega\Upsilon_0$ is equivalent to Υ (the proof is not straightforward though).

The follow theorem summarizes the results on the strength of $\Pi\Omega\Upsilon_k$ comparing with $\Pi\Omega_k$ and Υ, which is captured in Figure 1 and will be studied in Section 5. The key result is that $\Pi\Omega\Upsilon_k$ is strictly weaker than Υ for any $k \geq 1$, and as k increases, its strength is strictly weakened. Therefore, we found a new family of n classes of failure detectors that are all strictly weaker than Υ. It not only shows that Υ is not the weakest failure detector ever, but also suggests that there are still quite some room under Υ to fit in non-trivial failure detectors.

Theorem 3. *The followings hold regarding the strength of $\Pi\Omega\Upsilon_k$. (1) $\Pi\Omega\Upsilon_0 \equiv \Upsilon$. (2) $\Pi\Omega\Upsilon_k \prec \Pi\Omega\Upsilon_{k-1}$ for all $k \geq 1$. (3) $\Pi\Omega_j \not\preceq \Pi\Omega\Upsilon_k$ for all $1 \leq k \leq n$ and $1 \leq j \leq n$. (4) $\Pi\Omega\Upsilon_k \preceq \Pi\Omega_j$ for all $k \geq j \geq 1$. (5) $\Pi\Omega\Upsilon_k \not\preceq \Pi\Omega_j$ for all $j \geq k+2$ and $k \geq 1$.*

The algorithm that solves n-set agreement using $\Pi\Omega\Upsilon_k$ is based on the algorithm using Υ in [10], with modifications to (a) isolate the algorithm for each individual component; (b) obtain the size of each component; and (c) deal with the case that $S = \emptyset$ in the live component. The full algorithm and its proof are included in [5].

5 Comparing failure detectors

This section is the central place to show all the results captured in Figure 1 and stated in Theorems 1 and 3. Since Υ is already a very weak failure detector, one can imagine that it would be a subtle and delicate task to show that under Υ there are still such structure in which a series of failure detectors have various strengths. Indeed, besides those obvious transformations, other results on possible or impossible transformations are quite delicate and require subtle techniques to prove them (and a few of them are still open). These proofs really show the subtle relationship between the failure detectors. Unfortunately, due to the space constraint, we can only include the full proofs in [5]. To compensate, we provide intuitive ideas and proof outlines for some key proofs.

5.1 Possible transformations

For possible transformations, we need to prove all the arrows in Figure 1. Most transformations are obvious from the failure detector definitions.

Lemma 1. *(1) $\Pi\Omega_k \preceq \Pi\Omega_{k-1}$; (2) $\Pi\Omega\Upsilon_k \preceq \Pi\Omega\Upsilon_{k-1}$; (3) $\Pi\Omega_k \preceq \Omega_k$; (4) $\Pi\Omega\Upsilon_k \preceq \Upsilon$.*

Proof. The first two parts hold directly by the definition of failure detectors. The last two parts hold because we can treat Ω_k and Υ as a special case of partitioned failure detectors with only a single component P. □

Lemma 2. *$\Pi\Omega\Upsilon_k \preceq \Pi\Omega_k$ for all $k \geq 1$.*

Proof Outline. For the transformation from $\Pi\Omega_k$ to $\Pi\Omega\Upsilon_k$, the idea is for each component to come up with the set of at most *lbound* leaders, then the S output of $\Pi\Omega\Upsilon_k$ is the complement of the leader set with respect to the component, and *lbound* and *cid* outputs of $\Pi\Omega\Upsilon_k$ are copied from $\Pi\Omega_k$. The key is that for a live component, the leader set stabilizes and contains at least one correct process. Therefore, its complement S cannot be the set of correct processes. Moreover, if $S = \emptyset$, it means that all processes in the component are eventual leaders, in which case the *lbound* must be at least the number of correct processes in the component. The transformation still needs to solve the problem of estimating the membership of each component, which is addressed in the full transformation algorithm and its proof in [5]. □

Lemma 3. *(1) $\Omega_1 \preceq \Pi\Omega_1$; (2) $\Upsilon \preceq \Pi\Omega\Upsilon_0$*

The transformations for the above lemma are not straigthforward [5].

5.2 Impossible transformations

Proving the impossible transformations is the critical step to establish the results of this paper. For these proofs, it is sometimes convenient to view it as an adversary trying to defeat any possible transformations. The adversary can (a) see the current output generated by a transformation; (b) manipulate the outputs of the failure detector to be transformed; (c) schedule the executions of processes; and (d) crash processes to prevent the transformation from succeeding.

Among all the impossible transformations captured by the non-existent directed path in Figure 1, several of them are critical ones, meaning that their impossibility implies the rest impossible transformations. This is based on the fact that if we show that $\mathcal{C}_1 \not\succeq \mathcal{C}_2$, then for all $\mathcal{C}_3 \preceq \mathcal{C}_1$ and all $\mathcal{C}_4 \succeq \mathcal{C}_2$, we have $\mathcal{C}_3 \not\succeq \mathcal{C}_4$. The following lemma shows one such critical impossible transformations.

Lemma 4. *$\Pi\Omega_2$ cannot be transformed into Υ, i.e., $\Pi\Omega_2 \not\succeq \Upsilon$.*

Proof Outline. We know that Ω_n can be transformed to Υ easily by taking the complement of the Ω_n output. The reason that this transformation cannot be adapted to $\Pi\Omega_k$ is that $\Pi\Omega_k$ allows a live component P_j in which all processes are eventual leaders and *lbound* stabilizes to $|P_j|$. If we take the complement of the leader set in P_j with respect to P_j we get an empty set. The proof explores this basic idea.

In the case of $\Pi\Omega_2$, suppose for a contradiction that there is a transformation T from $\Pi\Omega_2$ to Υ. The adversary constructs a run in which the $\Pi\Omega_2$ has a partition $\pi = \{P_1, P_2\}$, where $P_1 = \{p\}$. It sets *lbound* of every process to 1 and p's *isLeader* always to *True*, making P_1 a live component of $\Pi\Omega_2$. It will manipulate the *isLeader* outputs for processes in P_2 to create a contradiction. Whenever the S output of Υ in P stabilizes to some subset S_i, the adversary suppresses all processes in $P \setminus S_i$ (i.e., prohibit these processes from taking any steps) for long enough time to force T to stabilize the S output to a different set $S_{i+1} \neq S_i$, because S_i appears to be the exact set of correct processes. Once T changes the S output, the adversary releases the suppressed processes so that they take some steps, and then it repeats the procedure for S_{i+1}, and so on. The adversary can keep doing so because $P \setminus S_i$ contains either p or some process in P_2, and thus it can always set *isLeader* of some process in $P \setminus S_i$ to *True* without violating the $\Pi\Omega_2$ requirement. The result is that the adversary forces T into an infinite run in which the S output never stabilizes, a contradiction. □

Lemma 4 implies that for all $\Pi\Omega_k$ with $k \geq 2$, $\Pi\Omega_k$ cannot be transformed into Υ. This is the first key result. Moreover, because $\Pi\Omega_k$ can be transformed into $\Pi\Omega\Upsilon_k$, Lemma 4 further implies that $\Pi\Omega\Upsilon_k$ is strictly weaker than Υ, the second key result of the paper. Next lemma shows another key result of the paper.

Lemma 5. *Υ cannot be transformed into $\Pi\Omega_n$ when $n \geq 2$.*

Proof Outline. Suppose there is a transformation T. If the partition of $\Pi\Omega_n$ generated by transformation T contains only a single component, then the proof is the same as proving Υ cannot be transformed into Ω_n in [10]. If the partition of $\Pi\Omega_n$ has at least two components, let P_1 be one of the components. The adversary first sets the Υ output to $P \setminus P_1$, and then repeatedly suppress the leader processes in all components that are potentially live components for $\Pi\Omega_n$ (these are called *quasi-live components* in the proofs), the purpose of which is to construct an infinite run in which there is no live component. The only way the transformation can counter this measure is by setting the *lbound* outputs of processes in P_1 to $|P_1|$. But the adversary can counter this again by crashing all processes in P_1, setting Υ output to P_1, and re-apply the suppression technique. The result is a run in which no live component exists. The key is that the adversary need to wait until the *lbound* output on P_1 is at least the size of a component to crash the component. This guarantees that the transformation cannot set *lbound* on $P \setminus P_1$ to $|P \setminus P_1|$ to defeat the adversary. $\square$

Lemmata 4 and 5 establish that Υ and $\Pi\Omega_k$ with $k \geq 2$ are not comparable. Together with the possible transformations of Lemma 2, they immediately imply that $\Pi\Omega\Upsilon_k$ is strictly weaker than both Υ and $\Pi\Omega_k$ for any $k \geq 2$.

Next lemma summarizes all other critical impossible transformations proven so far. The proofs to these results are technically involved and can be found in [5].

Lemma 6. *The following results hold: (1) $\Omega_k \not\preceq \Pi\Omega_{k-1}$ for any $k \geq 2$. (2) $\Pi\Omega\Upsilon_k \not\preceq \Pi\Omega\Upsilon_{k-1}$ for any $k \geq 1$. (3) $\Pi\Omega_{k+1} \not\preceq \Pi\Omega\Upsilon_{k-1}$ for any $k \geq 2$.*

In conclusion, Theorem 1 is implied by Lemma 1(1)(3), Lemma 3(1), Lemma 4, Lemma 5 and Lemma 6(1). Theorem 3 is implied by Lemma 1(2)(4), Lemma 3(2) and Lemma 6(2)(3).

There are still an open problem left before we can completely characterize all relationships in Figure 1. It is whether $\Pi\Omega_k$ can be transformed into $\Pi\Omega\Upsilon_{k-1}$ for any $k \geq 2$. We conjecture that this transformation is impossible. If so, Figure 1 is indeed a full characterization of all relationships.

6 Results in the Message-Passing Model

Partition approach can also be applied in the message-passing model to define weaker failure detectors for k-set agreement. We briefly summarize some of the results we obtained in the message-passing model. The complete results are included in [7].

In the message-passing model, it is shown in [17] that besides Ω_k a majority of correct processes is required to solve k-set agreement. The majority requirement can be generalized to the class of *quorum failure detectors* Σ defined in [8]: a failure detector in Σ outputs a set of processes called quorum such that: $(\Sigma 1)$ any two quorums intersect; and $(\Sigma 2)$ eventually all quorums contain only correct processes. Thus, we applied the partition approach to the class of failure detectors $\Omega_k \times \Sigma$ to define weaker failure detectors.[5]

We first applies static partitions to $\Omega_k \times \Sigma$ and define Π_k, which is similar to $\Pi\Omega_k$ but replacing the *cid* output with the quorum output. More specifically, the output of a failure detector $\mathcal{D}$ in Π_k for process p is a tuple $(\mathit{isLeader}, \mathit{lbound}, \mathit{Quorum})$, where *isLeader* is a Boolean value indicating whether this process is a leader, *lbound* is a non-negative integer indicating the upper bound on the number of possible leaders in

[5] Given two classes of failure detectors $\mathcal{C}_1$ and $\mathcal{C}_2$, class $\mathcal{C}_1 \times \mathcal{C}_2$ is the cross-product of the two, i.e., $\mathcal{C}_1 \times \mathcal{C}_2 = \{(\mathcal{D}_1, \mathcal{D}_2) \mid \mathcal{D}_1 \in \mathcal{C}_1, \mathcal{D}_2 \in \mathcal{C}_2\}$.

$$\begin{tikzcd}
\Omega_1 \times \Sigma \arrow[r, shift left] \arrow[d] & \Pi_1 \arrow[l, shift left] \arrow[r, shift left] \arrow[d] & \Pi_1^S \arrow[l, shift left] \arrow[d] \\
\Omega_2 \times \Sigma \arrow[r] \arrow[d, dash, dotted] & \Pi_2 \arrow[r] \arrow[d, dash, dotted] & \Pi_2^S \arrow[d, dash, dotted] \\
\Omega_{k-1} \times \Sigma \arrow[r] \arrow[d] & \Pi_{k-1} \arrow[r] \arrow[d] & \Pi_{k-1}^S \arrow[d] \\
\Omega_k \times \Sigma \arrow[r] & \Pi_k \arrow[r] & \Pi_k^S
\end{tikzcd}$$

Fig. 3. Relationship diagram for failure detectors in the message-passing model. All failure detector classes in the diagram can be used to solve k-set agreement ($n \geq 2k - 2$ is required to show that transformations from $\Omega_k \times \Sigma$ to Π_{k-1}^S and stronger classes are impossible).

p's partitioned component, and *Quorum* $\subseteq P$. A failure detector $\mathcal{D}$ is in the class Π_k if for any failure pattern F and any failure detector history $H \in \mathcal{D}(F)$, there exists a partition $\pi = \{P_1, \ldots, P_s\}$ of P, such that H satisfies the following set of safety and liveness properties. The safety properties are ($\Pi\Omega1$) as for $\Pi\Omega_k$ and the following two properties related to the quorum outputs:

($\Pi\Sigma1$) The quorum output of a process p is always contained within p's partitioned component. Formally, $\forall t \in \mathcal{T}, \forall p \notin F(t), H(p,t).\textit{Quorum} \subseteq \pi[p]$.

($\Pi\Sigma2$) The quorum outputs in the same partitioned component always intersect. Formally, $\forall t_1, t_2 \in \mathcal{T}, \forall p_1 \notin F(t_1), \forall p_2 \notin F(t_2), \pi[p_1] = \pi[p_2] \Rightarrow H(p_1,t_1).\textit{Quorum} \cap H(p_2,t_2).\textit{Quorum} \neq \emptyset$.

The liveness part specifies that there exists one partitioned component P_j such that the properties ($\Pi\Omega2$–5) of $\Pi\Omega_k$ hold plus the following:

($\Pi\Sigma3$) Eventually the quorum outputs by all processes in P_j contain only correct processes. Formally $\exists t_0 \in \mathcal{T}, \forall t \geq t_0, \forall p \in P_j \backslash F(t), H(p,t).\textit{Quorum} \subseteq \textit{correct}(F)$.

From the definition, we can see that Π_k follows the partition approach and is a static partitioning of $\Omega_k \times \Sigma$: each component P_i has a failure detector with all the safety properties of $\Omega_{k_i} \times \Sigma$ resticted to P_i where $k_i = \textit{lbound}(P_i)$ and $\sum k_i \leq k$, while at least one component P_j also satisfies all liveness properties of $\Omega_{k_j} \times \Sigma$.

Next we further weaken Π_k by allowing dynamic splitting of components during the run, which leads to the definition of Π_k^S. Failure detectors in Π_k^S output a tuple (*isLeader*, *lbound*, *Quorum*, *cid*). Informally, a failure detector in Π_k^S allows partitioned components to further split during the run, but it uses *cid* to differenciate different components and requires the quorum outputs in a component after the splitting intersects with all quorum outputs before the splitting. The formal definition is included in [7].

With the new families of failure detectors $\{\varPi_z\}_{1\le z\le k}$, and $\{\varPi_z^S\}_{1\le z\le k}$, we compare their strengths with $\{\varOmega_z \times \varSigma\}_{1\le z\le k}$. Based on a siginificant amount of proof work, we summarize their relationship with a nice lattice structure shown in Figure 3. Several important results are summarized by the lattice. First, as we expected $\varPi_k$ weakens $\varOmega_k \times \varSigma$,[6] and $\varPi_k^S$ further weakens $\varPi_k$ for all $k > 1$. Second, even failure detectors in $\varPi_2$ with just two components is not strong enough to be transformed into $\varOmega_k \times \varSigma$, and even failure detectors in $\varPi_2^S$ with only one dynamic split is not strong enough to be transformed into $\varPi_k$. This shows that partitioning and dynamic splitting are indeed efficient techniques that weaken failure detectors. Third, for all $z \ge 2$, none of the classes $\varOmega_z \times \varSigma$, $\varPi_z$, and $\varPi_z^S$ can be transformed into $\varOmega_{z-1} \times \varSigma$, $\varPi_{z-1}$, or $\varPi_{z-1}^S$. In fact, using a result in [17] we further show that $\varOmega_z \times \varSigma$, $\varPi_z$, and $\varPi_z^S$ are not strong enough to solve $(z-1)$-set agreement. In [7], we further show that the lattice structure in Figure 3 still holds (under certain mild assumptions) even if we assume that a majority of processes are correct in the system model.

Finally, we design a new algorithm in the message-passing model that solves k-set agreement using $\varPi_k^S$. The algorithm is based on the Paxos algorithm structure [15], but has significant new additions with much more complicated proofs to deal with the subtleties introduced by dynamic splittings of partitioned failure detectors.

7 Concluding Remarks

In [5] we further demonstrate the partition approach by defining a new failure detector $\varPi\varUpsilon$, which is the result of applying the approach directly to $\varUpsilon$. We show that $\varPi\varUpsilon$ is enough to solve n-set agreement but is strictly weaker than $\varUpsilon$. $\varPi\varUpsilon$ is stronger than $\varPi\varOmega\varUpsilon_{n-1}$ but is incomparable with $\varPi\varOmega\varUpsilon_k$ for $k \le n-2$.

We have shown that the partition approach is effective in weakening a number of failure detectors for k-set agreement. However, the partition approach proposed is still an informal method, and sometimes it requires ad-hoc adjustments. One future direction is to see how the approach and the partitioned failure detectors can be formally treated. In particular, it would be interesting to see if one could formally define a general class of partitioned failure detectors and define the weakest failure detectors among all partitioned failure detectors for k-set agreement.

The discovery of failure detectors even weaker than $\varUpsilon$ may suggest that the conjecture made in [10] that n-set agreement is the minimum decision task in terms of minimum information required might not be true. This is another research direction to see if there is any other decision task strictly weaker than n-set agreement in terms of failure information needed to solve the problem.

References

1. E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 91–100. ACM Press, May 1993.

[6] Actually, $\varPi_k$ weakens $\varSigma$ in all cases, and weakens $\varOmega_k$ in most cases.

2. T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722, July 1996.
3. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, Mar. 1996.
4. S. Chaudhuri. More *choices* allow more *faults*: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, July 1993.
5. W. Chen, Y. Chen, and J. Zhang. On failure detectors weaker than ever. Technical Report TR-2007-50, Microsoft Research, May 2007.
6. W. Chen, J. Zhang, Y. Chen, and X. Liu. Failure detectors and extended Paxos for k-set agreement. Technical Report TR-2007-48, Microsoft Research, May 2007.
7. W. Chen, J. Zhang, Y. Chen, and X. Liu. Partition approach to failure detectors for k-set agreement. Technical Report TR-2007-49, Microsoft Research, May 2007.
8. C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, and S. Toueg. The weakest failure detectors to solve certain fundamental problems in distributed computing. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing*, pages 338–346, July 2004.
9. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.
10. R. Guerraoui, M. Herlihy, P. Kouznetsov, N. Lynch, and C. Newport. On the weakest failure detector ever. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing*, Aug. 2007.
11. M. Herlihy and L. D. Penso. Tight bounds for k-set agreement with limited scope accuracy failure detectors. *Distributed Computing*, 18(2):157–166, 2005.
12. M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858–923, 1999.
13. M. P. Herlihy and J. M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Prog. Lang. Syst.*, 12(3):463–492, July 1990.
14. P. Jayanti. Robust wait-free hierarchies. *J. ACM*, 44(4):592–614, 1997.
15. L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.
16. A. Mostefaoui, S. Rajsbaum, and M. Raynal. The combined power of conditions and failure detectors to solve asynchronous set agreement. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing*, pages 179–188, July 2005.
17. A. Mostefaoui, S. Rajsbaum, M. Raynal, and C. Travers. Irreducibility and additivity of set agreement-oriented failure detector classes. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing*, pages 153–162, July 2006.
18. A. Mostefaoui and M. Raynal. k-set agreement with limited accuracy failure detectors. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing*, pages 143–152, July 2000.
19. M. Raynal and C. Travers. In search of the holy grail: Looking for the weakest failure detector for wait-free set agreement (Invited talk). In *Proc. 10th Int'l Conference On Principles Of Distributed Systems (OPODIS'06)*, pages 1–17, Dec. 2006.
20. M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing*, 29(5):1449–1483, 2000.
21. J. Yang, G. Neiger, and E. Gafni. Structured derivations of consensus algorithms for failure detectors. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing*, pages 297–306, June 1998.