On *k*-Set Consensus Problems in Asynchronous Systems

Roberto De Prisco, Dahlia Malkhi, and Michael Reiter

Abstract—In this paper, we investigate the *k*-set consensus problem in asynchronous distributed systems. In this problem, each participating process begins the protocol with an input value and by the end of the protocol must decide on one value so that at most *k* total values are decided by all correct processes. We extend previous work by exploring several variations of the problem definition and model, including for the first time investigation of Byzantine failures. We show that the precise definition of the validity requirement, which characterizes what decision values are allowed as a function of the input values and whether failures occur, is crucial to the solvability of the problem. For example, we show that allowing default decisions in case of failures makes the problem solvable for most values of *k* despite a minority of failures, even in face of the most severe type of failures (Byzantine). We introduce six validity conditions for this problem (all considered in various contexts in the literature), and demarcate the line between possible and impossible for each case. In many cases, this line is different from the one of the originally studied *k*-set consensus problem.

Index Terms—Agreement problems, Byzantine failures, consensus, crash failures, distributed systems, validity conditions.

1 Introduction

The consensus problem is an abstraction of many coordination problems in a distributed system that can suffer process failures. Roughly speaking, the consensus problem is to have processes of a distributed system agree on a common decision. Because of the many practical problems that can be reduced to this simple primitive, consensus has been thoroughly studied. We refer the reader to [25] for a detailed discussion of consensus.

A fundamental result about consensus is that there is no deterministic algorithm for solving it in asynchronous distributed systems subject to process failures [17], [24]. Spurred by this result, much research has focused on demarcating the line between possible and impossible by varying the problem on a number of axes, such as by imposing some degree of synchrony on the system (e.g., [3], [15], [16], [12]) or by weakening the liveness (e.g., [8], [10], [29]) or safety conditions of consensus itself. A work of the latter type is *k*-set consensus [13], which weakens the safety conditions of consensus to allow the set of process decision values to be of cardinality up to $k > 1$ (versus $k = 1$ for consensus). It is known that to solve *k*-set consensus in an asynchronous system subject to at most t process failures, the condition $t < k$ is both sufficient [13] and necessary [9], [20], [30], [18], [6], [1], [19].

Besides requiring processes to agree on at most *k* decision values, *k*-set consensus (and consensus when $k = 1$) imposes a "validity" condition that expresses the allowable set of decision values as some function of process' input values. For example, the validity condition adopted for *k*-set consensus requires that each correct process decide on a value that is equal to some process' input value [13]. While consensus remains impossible for any nontrivial validity condition (i.e., that allows for two possible decision values) [17], [24], in this paper, we show for the first time that the validity condition has a profound impact on when *k*-set consensus is solvable.

More precisely, in this paper we provide a systematic investigation of the impact of the validity condition on the solvability of the *k*-set consensus problem in asynchronous distributed systems. Our investigation includes six variations of the validity condition, most of which have appeared before in the literature on consensus or related problems. Moreover, we explore these validity conditions in the context of four different asynchronous models of distributed computing, as characterized by the following two axes:

- **Type of process failures**: We consider two models of process failures, namely crash and Byzantine. Prior work has considered *k*-set consensus only in the context of crash failures; our work is the first exploration of this problem for Byzantine failures.
- **Communication model**: We consider two models of communication between processes. In the first, processes communicate by sending messages to one another over a completely connected network. In the second, processes communicate by modifying shared memory.

Thus, we consider the *k*-set consensus problem in 24 variations (four models $\times$ six validity conditions). In several cases, we completely characterize solvability. In some, we characterize solvability with very little uncertainty (i.e., a small gap between computable and impossible), and in a few cases we leave a substantial gap. The

- *R. De Prisco is with the MIT Laboratory for Computer Science, 545 Technology Sq., NE43-368, Cambridge, MA 02139 and Dipartimento di Informatic ed Applicazioni Università di Salerno, 84081 Baronissi (SA), Italy. E-mail: robdep@theory.lcs.mit.edu.*
- *D. Malkhi is with AT&T Labs-Research, 180 Park Ave., Florham Park, NJ 07932. E-mail: dalia@research.att.com.*
- *M. Reiter is with Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. E-mail: reiter@research.bell-labs.com.*

Manuscript received 3 Aug. 1999; revised 28 July 2000; accepted 1 Aug. 2000. For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 110354.

main lesson to take from our results is that unlike for consensus, the solvability of k-set consensus is very sensitive to the particular validity condition used.

The rest of this paper is structured as follows: In Section 2, we define the problem. We study the k-set consensus problem for message passing systems in Section 3. Section 4 presents the results for shared memory systems. Section 5 concludes and raises open problems.

2 The Problem

We consider a distributed system consisting of n processes denoted by $p_1, p_2, \ldots, p_n$. A process that follows its algorithmic specification throughout an execution is said to be *correct*, and a process that departs from its specification is said to be *faulty*. In a *crash model*, faulty processes are allowed to prematurely halt execution only. A correct process executes infinitely many instructions (at most one of which is the designated "decide" instruction). A faulty process executes only finitely many instructions. In a *Byzantine* model, a faulty process can deviate from its program arbitrarily. We assume that at most t processes fail, where $t \geq 1$ is a known, positive integer. We study two communication models, asynchronous message passing and asynchronous shared memory. Their precise specification is given below in Sections 3 and 4, respectively. We will use the following shorthands: *MP/CR* to denote the message-passing crash model, *MP/Byz* to denote the message-passing Byzantine model, *SM/CR* to denote the shared memory crash model and *SM/Byz* to denote the shared memory Byzantine model.

For any k, $1 \leq k \leq n$, we denote a k-set consensus problem by $\mathcal{SC}(k)$ or simply $\mathcal{SC}$ when k is not relevant. The $\mathcal{SC}(k)$problem is defined as follows. Each process p_i starts the computation with an input value v_i. We allow the set of possible input values from which p_i chooses v_i to be unconstrained, and in particular to have cardinality of size n or larger. Each correct process has to irreversibly "decide" on a value in such a way that three conditions, called *termination*, *agreement*, and *validity*, hold. These conditions are:

- **Termination:** Every correct process eventually decides.
- **Agreement:** The set of values decided by correct processes has size at most k.
- **Validity:** One of the following conditions.
  - **SV1** (strong V1): The decision of any correct process is equal to the input of some correct process.
  - **SV2** (strong V2): If all correct processes start with v then correct processes decide v.
  - **RV1** (regular V1): The decision of any correct process is equal to the input of some process.
  - **RV2** (regular V2): If all processes start with v then correct processes decide v.
  - **WV1** (weak V1): If there are no failures, then the decision of any process is equal to the input of some process.

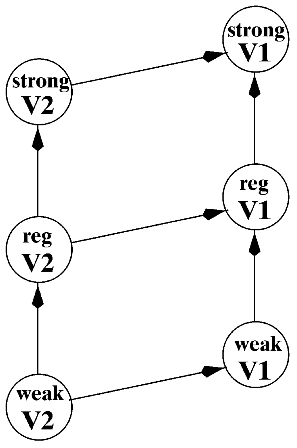

Fig. 1. Validity conditions. An arrow from a validity condition C to a validity condition D means that $\mathcal{SC}(C)$ is weaker than $\mathcal{SC}(D)$ is stronger than $\mathcal{SC}(C)$.

  - **WV2** (weak V2): If there are no failures and all processes start with v, then the decision of any process is equal to v.

Given a validity condition C, we denote by $\mathcal{SC}(k, C)$ the $\mathcal{SC}(k)$ problem defined with validity C. We also use the notation $\mathcal{SC}(C)$ if k is not relevant. We use the notation $\mathcal{SC}(k, t)$ to denote a $\mathcal{SC}(k)$ consensus problem with at most t failures allowed. The notation $\mathcal{SC}(k, t, C)$ denotes $\mathcal{SC}(k, t)$ with validity C.

We define a partial order on the $\mathcal{SC}$ problems based on the strength of the validity conditions. We say that $\mathcal{SC}(C)$ is *weaker* than $\mathcal{SC}(D)$ if the validity condition of $\mathcal{SC}(C)$ is logically implied by the validity condition of $\mathcal{SC}(D)$. If $\mathcal{SC}(C)$ is weaker than $\mathcal{SC}(D)$, then any run of a protocol that solves $\mathcal{SC}(D)$ also solves $\mathcal{SC}(C)$. Clearly, this also implies that any impossibility result that holds for $\mathcal{SC}(C)$ holds also for $\mathcal{SC}(D)$. Conversely, we say that $\mathcal{SC}(C)$ is *stronger* than $\mathcal{SC}(D)$ if $\mathcal{SC}(D)$ is weaker than $\mathcal{SC}(C)$. Fig. 1 shows the "weaker than" relation between the validity conditions.

$\mathcal{SC}(k, \text{RV}1)$ is the consensus problem as considered by Chaudhuri [13]. $\mathcal{SC}(1, \text{RV}1)$ and $\mathcal{SC}(1, \text{RV}2)$ are classical consensus problems (see, e.g., [25, chapter 6]. $\mathcal{SC}(1, \text{SV}2)$ has been considered in the Byzantine setting [23], [28]. $\mathcal{SC}(1, \text{WV}2)$ is a weak Byzantine agreement [21].

It is well-known that the case $k = 1$ cannot be solved for any nontrivial validity condition and, in particular, for any of the validity conditions that we consider here, or for any $t \geq 1$, both in the MP/CR model [17] and in the SM/CR model [24]. On the other hand, if $k = n$, then $\mathcal{SC}(k)$ is trivially solvable (each process decides its own value), even in the Byzantine setting, for any t and with the strongest validity condition we are considering, that is, validity SV1. Thus, we will henceforth be concerned only for the cases $2 \leq k \leq n - 1$. Since the problem is easily solvable for $t = 0$, we also assume that $t \geq 1$.

2.1 Summary of Results and Discussion

We have studied the six variations of the validity conditions presented in the previous section in four different models (MP/CR, MP/Byz, SM/CR, and SM/Byz). The results are summarized in four figures, one for each of the models that we consider. In particular, Fig. 2 summarizes the results for

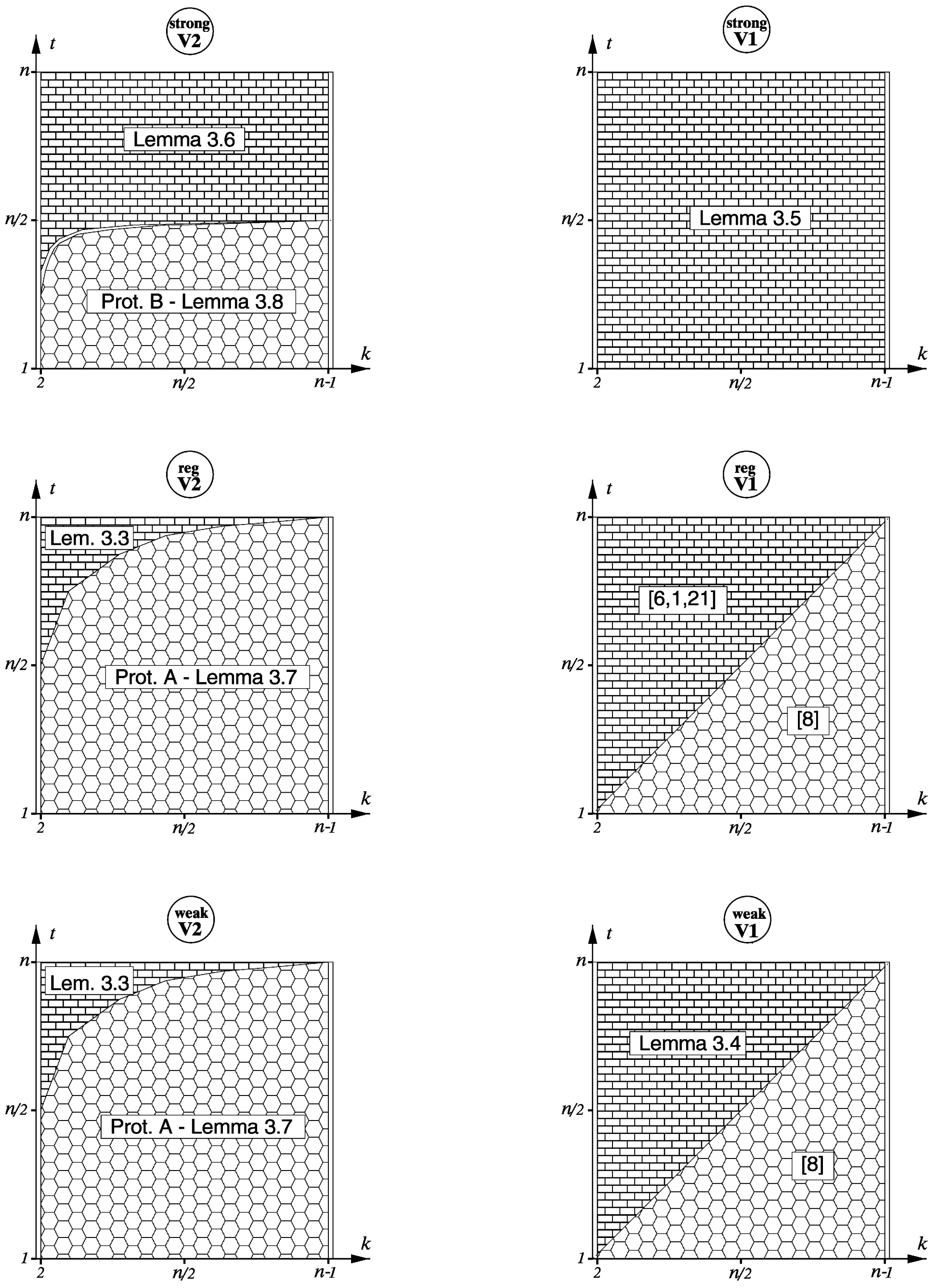

Fig. 2. Message-passing crash model (MP/CR). Regions filled in brick pattern indicate impossibility. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn for the case $n = 64$ processes.

the MP/CR model. Fig. 4 summarizes the results for the MP/Byz model. Fig. 5 summarizes the results for the SM/CR model and Fig. 6 summarizes the results for the SM/Byz model. These figures appear in the corresponding sections. Each figure contains six graphs, one for each validity condition, showing the solvability region and the impossibility region. Each region in the graphs has a label that provides a reference to the lemma proving the corresponding result where the reader can find the mathematical definition of the solvability or impossibility region.

The rest of the paper will provide the proofs of these results, beginning with the proofs for message passing models in Section 3 and then moving to shared memory models in Section 4. Because our message passing models and shared memory models are standard, in a few cases we can employ previous results relating the two models in the derivation of our results. In particular, because there are known translations from any message passing algorithm to a shared memory algorithm for the same problem [9], [26] (see also [7] for a more accessible description of these results), in some cases, we can use the algorithms we develop for message passing models (Section 3) in shared memory models (Section 4). It is also the case that impossibilities in the shared memory model apply in the message-passing models. We exploit this fact by utilizing

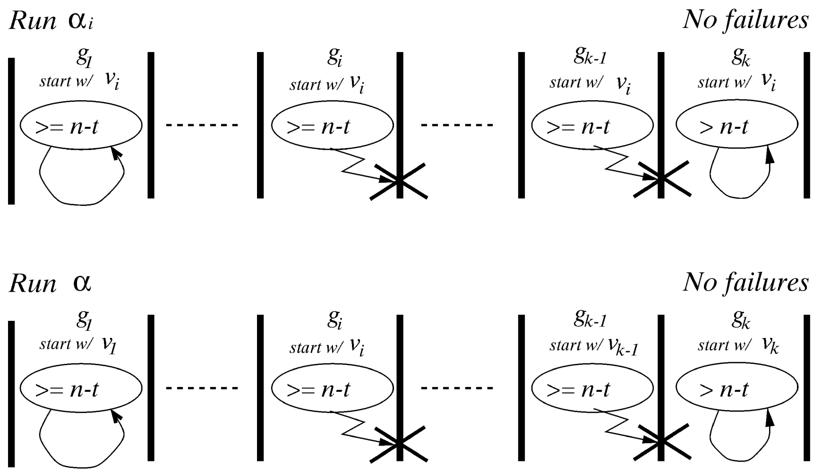

Fig. 3. The run of Proof 3.3.

known impossibilities for k-set consensus in the shared memory model [9], [20], [30], [6], [1] to derive negative results in the message passing model. To avoid intermixing our treatment of message passing and shared memory models, however, in some cases we will directly prove negative results for message passing models in Section 3 even though a more general result is proven in Section 4.

A benefit of utilizing previous results in our impossibility proofs is that we can avoid the complex topological arguments that have characterized most prior impossibility proofs in this area [9], [20], [30]. Rather, in many cases we show that known impossibilities imply impossibilities for the new problem variants that we consider.

3 Message Passing Models

In this section, we consider $\mathcal{SC}$ in a message-passing model. Processes communicate by sending messages. We assume that the underlying communication network is complete, that is, there is a communication channel for each pair of processes. Moreover, communication is reliable: messages are not lost, duplicated, forged, or generated by the network. Processes may take an arbitrary (but finite) time to execute a step and message delivery can have an arbitrary (but finite) delay over the communication network. That is, the system is *asynchronous*.

3.1 Message Passing Model with Crash Failures

In this section, we consider the message-passing crash (MP/CR) model. As noted in Section 1, for these systems we already know the line between computable and impossible for $\mathcal{SC}(k, t, \text{RV1})$:

Lemma 3.1 ([13]). *In the message-passing crash model (MP/CR), there is a protocol for $\mathcal{SC}(k, t, \text{RV1})$, for $t < k$.*

Lemma 3.2 ([9], [20], [30]). *In the crash models, there is no protocol for $\mathcal{SC}(k, t, \text{RV1})$, for $t \geq k$.*

We remark that Lemma 3.2 holds for both MP/CR and SM/CR models.

By Lemma 3.1, we know that $\mathcal{SC}(k, t, \text{RV2})$, $\mathcal{SC}(k, t, \text{WV1})$, and $\mathcal{SC}(k, t, \text{WV2})$ are solvable for $t < k$ because these $\mathcal{SC}$ problems are weaker than $\mathcal{SC}(k, t, \text{RV1})$. By Lemma 3.2, $\mathcal{SC}(k, t, \text{SV1})$ cannot be solved for $t \geq k$ because $\mathcal{SC}(k, t, \text{SV1})$ is stronger than $\mathcal{SC}(k, t, \text{RV1})$.

In Sections 3.1.1 and 3.1.2, we provide further impossibility results and protocols, respectively. Fig. 2 shows a graphical representation of the results provided in this section.

For $\mathcal{SC}(\text{RV2})$ and $\mathcal{SC}(\text{WV2})$, there is a very tiny gap between our possibility and impossibility results (Lemmas 3.3 and 3.7), formed by the cases where n is a multiple of k. These are isolated points on the line that separates possible from impossible. For $\mathcal{SC}(\text{SV2})$ there is also small gap between our possibility and impossibility results (Lemmas 3.6 and 3.8).

3.1.1 Impossibilities

In this section, we provide impossibility results for the MP/CR model. An ingredient in most of our impossibility results is the fact that in any protocol tolerating t failures, a process must be able to decide after communicating with at most $n - t$ processes (including itself). Indeed, if a process waited to communicate with more than $n - t$ processes, termination could not be achieved: the runs in which there were exactly t faulty processes that do not send any messages, would not terminate.

Lemma 3.3. *In the MP/CR model, there is no protocol for $\mathcal{SC}(k, t, \text{WV2})$, for $t \geq \frac{(k-1)n+1}{k}$.*

Proof. For a contradiction, assume that such a protocol A exists (see Fig. 3). In the rest of the proof, we use the notation $\mathcal{SC}_P(k, t, C)$ to explicitly state the set P of processes among which k-set consensus is to be solved. Denoting by $\mathcal{P}$ the set of all processes, we have that A solves $\mathcal{SC}(k, t, \text{WV2})$.

Since $t \geq ((k-1)n + 1)/k$ implies $n \geq k(n - t) + 1$, we can partition the n processes into k groups $g_1, g_2, \ldots, g_k$ of disjoint processes with $g_1, , \ldots, g_{k-1}$ containing exactly $n - t$ processes and g_k containing at least $n - t + 1$ processes. If $t = n$, we let $g_1, g_2, \ldots, g_{k-1}$ be singleton sets of processes and we let g_k contain at least two processes (this is possible because we only consider $k < n$).

First, we claim that there is a run of A where only processes in g_k take steps and such that two values are decided. To see why, assume that all the runs involving only processes of g_k are such that only one value is decided. Then we could use A to solve $\mathcal{SC}_{g_k}(1, 1, \mathrm{WV2})$: g_k contains at least $n - t + 1$ processes, so that even if one of them is faulty, we still have at least $n - t$ correct processes in g_k, and hence the protocol has to terminate. Moreover, validity WV2 is ensured because any run of A on g_k in which all processes in g_k begin with the same input value is indistinguishable to processes in g_k from a run of A on $\mathcal{P}$ in which all processes in $\mathcal{P}$ begin with that input value but processes in g_k receive no messages from processes outside g_k before deciding. However, this contradicts [17], since no such protocol can exist. Hence, there is a run α_k in which only processes in g_k take steps and they decide on at least two different values, say v_k, v_{k+1}. Let $v_1, \ldots, v_{k-1}$ be $k - 1$ values different from v_k, v_{k+1}.

Fix i, $i \in \{1, 2, \ldots, k - 1\}$ and consider a run α_i constructed as follows: All processes are correct, all start with v_i, and all messages sent to processes in g_j, $j = 1, 2, \ldots, k$ by processes not in g_j are delayed until all processes in g_j make a decision (they eventually make a decision because g_j contains at least $n - t$ processes and they are all correct). Since we had assumed that A can solve $\mathcal{SC}_{\mathcal{P}}(k, t, \mathrm{WV2})$, we have by the validity condition WV2 that all processes, in particular those in group g_i, decide v_i.

Now consider a run α constructed as follows: All processes are correct, for each i, $i = 1, 2, \ldots, k - 1$, every process in g_i starts with v_i and processes in g_k start with the same values they start in α_k. Moreover, for each i, $i = 1, 2, \ldots, k$, all messages sent to processes in group g_i by processes not in g_i are delayed until all processes in g_i have decided (they eventually decide because there are $n - t$ processes in g_i and all of them are correct). We can use A to solve $\mathcal{SC}_{\mathcal{P}}(k, t, \mathrm{WV2})$ in α. However, for each i, $i = 1, 2, \ldots, k$, processes in g_i cannot distinguish between run α_i and run α. Indeed, in both runs they only communicate with processes in g_i before making a decision and in both runs processes in g_i start with the same value. Since, for $i = 1, 2, \ldots, k - 1$, in run α_i processes in g_i decide v_i, they must decide v_i also in α. Since in run α_k processes in g_k decide on v_k and v_{k+1}, they must decide v_k and v_{k+1} also in α. Hence, we have that $k + 1$ values are decided in α. Thus, the agreement condition is violated and this contradicts the hypothesis that A solves $\mathcal{SC}_{\mathcal{P}}(k, t, \mathrm{WV2})$. □

Lemma 3.4. *In the MP/CR model, there is no protocol for* $\mathcal{SC}(k, t, \mathrm{WV1})$, *for* $t \geq k$.

Proof. For a contradiction assume that there exists such a protocol A. We claim that A can be used to solve $\mathcal{SC}(k, t, \mathrm{RV1})$ for $t \geq k$. To see why, consider any run α in which $f \leq t$ processes are faulty and let g be the set of correct processes and g' be the set of faulty processes. Now consider a run α' that is identical to α except that all processes are correct and any message sent by any $p \in g'$ in α' after the time that p failed in α is delayed until after all processes in g decide. That is, for each $p_i \in g$ and each $p_j \in g'$, p_i receives a message from p_j at time T in α' iff p_i receives the same message at time T from p_j in α. By the validity condition WV1, each process decides on some process' input in α'. Clearly, processes in g cannot distinguish between α and α'. Hence, processes in g decide the same value in α as they decide in α', and so validity RV1 is satisfied in α. In other words, protocol A solves $\mathcal{SC}(k, t, \mathrm{RV1})$ for $t \geq k$, contradicting Lemma 3.2. □

Lemma 3.5. *In the MP/CR model, there is no protocol for* $\mathcal{SC}(k, t, \mathrm{SV1})$.

Proof. For a contradiction, assume that there exists such a protocol A. Let α be an execution of A in which all processes are correct and they all start with different values. Let v a decision made by at least two processes (there is always such a decision since $k < n$). Because of validity SV1, v is the input of some process p_i, and since all inputs are different only p_i has v as input. Now con– sider the run α' that is the same as α except that process p_i fails right after sending its last message. Clearly α and α' are indistinguishable, and thus each process (maybe with the exception of p_i) makes the same decision in both runs. Hence, in α' value v is decided by at least one process p_j, $j \neq i$. But only p_i has v as input and p_i is not correct in α', and so validity SV1 is violated. □

Lemma 3.6 *In the MP/CR model, there is no protocol for* $\mathcal{SC}(k, t, \mathrm{SV2})$, *for* $t \geq \frac{k}{2k+1} n$.

Proof. For a contradiction assume that there exists such a protocol A. Consider first the case $t \geq \frac{n}{2}$. Partition the system into two non-intersecting sets of processes, g, g', each containing at least $n - t$ processes (e.g., $|g| = |g'| = n/2$). This is always possible because $t \geq n/2$. Let α be a run of A in which all processes are correct, all start with different initial values denoted $v_1, v_2, \ldots, v_n$, and all communication between g and g' is delayed until after the decisions are made. We claim that n values are decided in α. To see this, fix any process $p_i \in g$, and consider a run α_i constructed as follows. The processes in g start with the same values as in α, and all except p_i crash after p_i reaches a decision. All the processes in g' start with v_i, but communication between g and g' is delayed until after p_i makes a decision. By SV2, p_i must decide v_i in α_i, and by indistinguishability of α from α_i, p_i must decide v_i in α. Similarly, runs α'_i can be constructed for every process $p'_i \in g'$, and hence, all processes must decide their own values in α. This contradicts the hypothesis that A solves the problem (for $k < n$).

Now consider the case $t < \frac{n}{2}$. In this case, $n - 2t > 0$ and the condition $t \geq n\frac{k}{2k+1}$ is equivalent to $k \leq \frac{n-t}{n-2t} - 1$. Let g be a subset of the system containing $n - t$ processes, and let $g_1, \ldots, g_{\lfloor \frac{n-t}{n-2t} \rfloor}$ be a partition of g into disjoint sets of size at least $n - 2t$ each. Let α be a run of A in which all the processes are correct, communication between g and the rest of the system is delayed until after all processes have decided and, for each i, processes in g_i start with a distinct value v_i. Fix i, and let $p_i \in g_i$ be some process. Consider a run α_i of A as follows: Processes in g_i are correct, all processes in $g \setminus g_i$ are faulty, and crash after p_i decides. All communication between g and the rest of the system is delayed until after p_i decides. By SV2, p_i must decide v_i, but since α is indistinguishable to p_i from α_i, p_i

must decide v_i in α. Therefore, in α, at least $\lfloor\frac{n-t}{n-2t}\rfloor$ different values are decided on. This contradicts the hypothesis that A solves the problem since $k \le \frac{n-t}{n-2t} - 1 < \lfloor\frac{n-t}{n-2t}\rfloor$. □

3.1.2 Protocols

In this section, we provide two protocols for the MP/CR model.

PROTOCOL A. Each process broadcasts its input and waits for $n-t$ messages. If all $n-t$ messages contain the same value v, then the process decides v, else it decides a default value v_0.

Lemma 3.7. *PROTOCOL A solves $\mathcal{SC}(k, t, \text{RV}2)$ in the MP/CR model for $t < \frac{k-1}{k}n$.*

Proof. We start by proving termination. The number of actual failures is less or equal to t. Hence, there are at least $n-t$ correct processes. Thus, each correct process eventually receives at least $n-t$ messages and is able to make a decision.

Now we prove agreement. By the sake of contradiction, assume that $k+1$ values are decided. One of them could be the default value, but at least k values, different from the default value, are decided. By the protocol, it is necessary that there be k disjoint sets $g_1, g_2, \ldots, g_k$, each consisting of at least $n-t$ processes such that each process in g_i sends a value v_i (with $v_i \neq v_j$ for $i \neq j$). Hence, there must be at least $k(n-t)$ processes. However, since $t < \frac{k-1}{k}n$, we have that $n-t > n/k$, and that $k(n-t) > n$, which implies that there must be more than n processes. This is impossible, since we have n processes.

Finally, we prove validity. Assume that all processes start with value v. Clearly, a process cannot receive two different values, since v is the only value being sent. Hence, by the protocol, each process that makes a decision decides v. □

PROTOCOL B. Each process broadcasts its input and waits for $n-t$ messages. One of these $n-t$ messages is the process' own message. If $n-2t$ messages contain the same value as its own, say v, the process decides v, else it decides a default value v_0.

Lemma 3.8. *PROTOCOL B solves $\mathcal{SC}(k, t, \text{SV}2)$ in the MP/CR model for $t < \frac{k-1}{2k}n$.*

Proof. We start by proving termination. The number of actual failures is less than or equal to t. Hence, there are at least $n-t$ correct processes. Thus, each correct process eventually receives at least $n-t$ messages and is able to make a decision.

Now we prove agreement. For the sake of contradiction, assume that $k+1$ values are decided. One of them could be the default value, but at least k values, different from the default value, are decided. By the protocol it is necessary that there be k disjoint sets $g_1, g_2, \ldots, g_k$, each consisting of at least $n-2t$ processes such that each process in g_i sends a value v_i (with $v_i \neq v_j$ for $i \neq j$). Hence, there must be at least $k(n-2t)$ processes. However, since $t < \frac{k-1}{2k}n$, we have that $k(n-2t) > n$, which implies that there must be more than n processes. This is impossible, since we have n processes.

Finally, we prove validity. Assume that all correct processes start with value v. We have to prove that a correct process decides v. Let p be a correct process. First, we observe that since p starts with v it decides v or v_0. Hence, it suffices to prove that p receives at least $n-2t$ messages with v. Among the $n-t$ messages p receives, at least $n-2t$ are from correct pro– cesses. Hence, process p receives at least $n-2t$ messages with v. □

3.2 Message Passing Model with Byzantine Failures

In this section, we consider the message-passing Byzantine (MP/Byz) model. In Section 3.2.1, we are concerned with impossibilities, and in Section 3.2.2, we provide protocols. Fig. 4 shows a graphical representation of the results.

For the MP/Byz model, the impossibility results and protocols we have provided in this section leave a small gap for the $\mathcal{SC}$ problem defined with validities WV2, RV2, and SV2, and a substantial gap for $\mathcal{SC}(\text{WV}1)$.

3.2.1 Impossibilities

In this section, we provide impossibility results for the MP/Byz model. Clearly, the impossibilities proved for the MP/CR model still hold. In particular, the impossibilities for $\mathcal{SC}(\text{SV}1)$ and $\mathcal{SC}(\text{WV}1)$ are directly derived from the corresponding ones for the MP/CR model. Next, we provide additional impossibilities.

Lemma 3.9. *In the MP/Byz model, there is no protocol that solves $\mathcal{SC}(k, t, \text{WV}2$, for $t \ge \frac{k}{2k+1}n$ and $t \ge k$.*

Proof. For a contradiction, assume that such a protocol A exists. We distinguish two cases: 1) $t \ge n/2$ and 2) $t < n/2$.

Consider case 1). Let $v_1, v_2, \ldots, v_{t+1}$ be $t+1$ different values. Let α be a run of A constructed as follows: The number of actual failures in α is $f = n-t-1$. Let F be the set of faulty processes and let $p_1, \ldots p_{t+1}$ be the correct processes. Process p_i has input v_i, for $i = 1, 2, \ldots, t+1$. Messages between any two correct processes are delayed until all correct processes decide, that is, correct processes communicate only with processes in F.

We now show that at least $k+1$ values are decided in α, which contradicts the hypothesis that A solves the problem. For each $i = 1, 2, \ldots, t+1$, consider a run α_i constructed as follows: All processes are correct, all have input v_i, and messages between processes not belonging to F are delayed until all processes not in F decide. By validity WV2, we have that in α_i all processes must decide v_i. Process p_i, for $i = 1, 2, \ldots, t+1$, cannot distinguish between α and α_i, if in α, the members of F behave as if they were correct and had v_i initially. Hence, p_i has to decide the same value in both runs. We have that process p_i decides v_i also in α. Since $v_1, v_2, \ldots, v_{t+1}$ are different, we have that $t+1$ values are decided in α. But $t \ge k$, hence at least $k+1$ values are decided in α.

Consider case 2). Since $t < n/2$, we have that $n-2t > 0$, and thus the condition $t \ge \frac{k}{2k+1}n$ is equivalent to $\frac{n-t}{n-2t} \ge k+1$. Then, we can partition the processes into

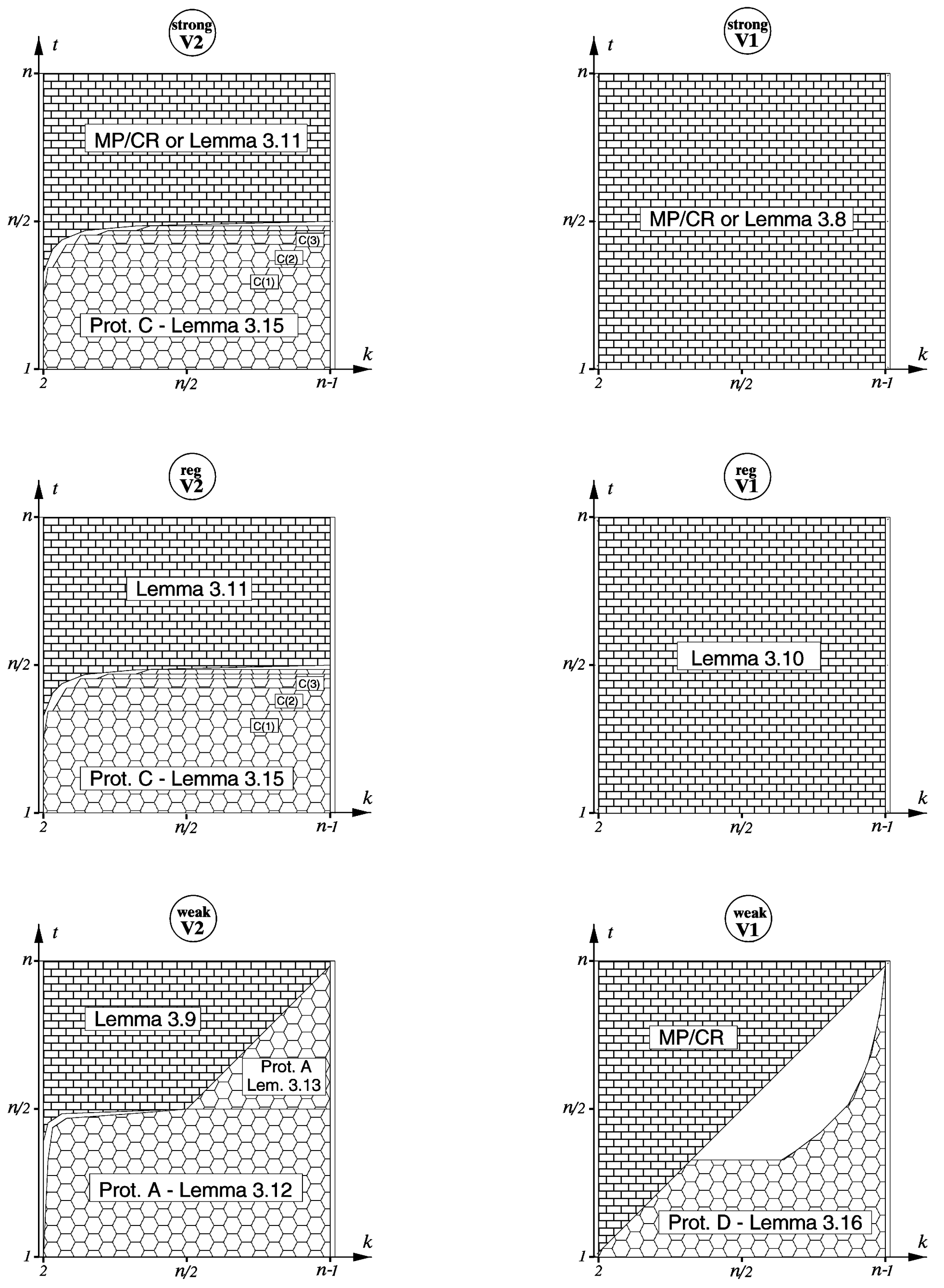

Fig. 4. Message-passing Byzantine model (MP/Byz). Regions filled in brick pattern indicate impossibility. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn for the case $n = 64$ processes.

$k+2$ groups, the first $k+1$ of which, denoted $g_1, g_2, \ldots, g_{k+1}$, each consists of at least $n-2t$ processes, and the last of which, denoted F, consists of t processes. Let α be a run of A constructed as follows. Let $v_1, v_2, \ldots, v_{k+1}$ be $k+1$ different values. Processes in g_i start with v_i, for $i = 1, 2, \ldots, k+1$, and processes in F are faulty. Processes in group g_i communicate only within g_i and with processes in F. For each group g_i, processes in F behave as correct processes with input v_i.

We now show that at least $k+1$ values are decided in α, which contradicts the hypothesis that A solves the problem. For each $i = 1, 2, \ldots, k+1$ consider a run α_i constructed as follows: All processes are correct, all have input v_i, and processes in group g_i communicate only within g_i and with processes in F. By validity WV2, we have that in α_i, all processes must decide v_i. Processes in g_i, for $i = 1, 2, \ldots, k+1$, cannot distinguish between α and α_i.

Hence, they have to decide the same value in both runs, and so processes in g_i decide v_i also in α. Since $v_1, v_2, \ldots, v_{k+1}$ are different, we have that $k+1$ values are decided in α. □

Lemma 3.10. *In the MP/Byz model, there is no protocol that solves $\mathcal{SC}(k, t, \text{RV}1)$.*

Proof. For a contradiction, assume that such a protocol A exists. Let α_1 be a run of A in which all processes are correct and each start with a different input value. Let

$v_1, \ldots, v_z$ be the set of values decided by correct processes. Because A satisfies validity RV1, each of the v_i is the input of some process. Since $z \leq k < n$, we have that there exists a value v_i, $1 \leq i \leq z$, decided by at least two processes, say p_1 and p_2.

Let process q be the process whose input in α_1 is v_i. Use A in the run α_2 in which q is faulty but behaves as in α_1, claiming that v_i is its input, but that it has v_i' as its input, with v_i' different from v_i and also from any other input. Since correct processes cannot distinguish between α_1 and α_2, they have to decide on the same value. We now distinguish two possible cases: 1) q is different from both p_1 and p_2, or 2) q is p_1 or p_2. If q is different from both p_1 and p_2, then both p_1 and p_2 are correct and thus they decide on v_i in α_2. However, v_i is not an input value in α_2. Hence, validity is violated. If q is p_1 (respectively, p_2) then p_2 (respectively, p_1) is correct and thus decides v_i in α_2. However, v_i is not an input value in α_2. Hence, validity RV1 is violated. This contradicts the hypothesis that A solves $\mathcal{SC}(k, t, \text{RV}1)$. □

Lemma 3.11. *In the MP/Byz model, there is no protocol for $\mathcal{SC}(k, t, \text{RV}2)$, for $t \geq \frac{k}{2(k+1)} n$.*

Proof. (Similar to Lemma 3.6.) For a contradiction, assume that such a protocol A exists. We distinguish two cases: 1) $t < n/2$ and 2) $t \geq n/2$.

Consider case 1. Since $t < n/2$, we have that $n - 2t > 0$, and thus, the condition $t \geq \frac{k}{2(k+1)} n$ is equivalent to $\frac{n}{n-2t} \geq k + 1$. Then, we can partition the processes in $k + 1$ groups each consisting of at least $n - 2t$ processes. Consider case 2. In this case, we partition the processes in $k + 1$ groups each consisting of at least one process.

In both cases, let $g_1, g_2, \ldots, g_k, g_{k+1}$ be the $k + 1$ groups of processes. Let $v_1, \ldots v_{k+1}$ be $k + 1$ different values and consider a run α constructed as follows. All processes are correct, processes in group g_i start with v_i. For each group g_i, there is a set of t processes not belonging to g_i, call it F_i, such that, for each i, communication is allowed only among processes in g_i and F_i until all processes have decided. Notice that the cardinality of $g_i \cup F_i$ is at least $n - t$ in both cases.

We now show that $k + 1$ values are decided in α, which contradicts the hypothesis that A solves the problem. Fix i, $1 \leq i \leq k + 1$, and consider run α_i. There are exactly t faulty processes and these processes are those in F_i. Processes in g_i are correct. All processes start with v_i. Faulty processes behave exactly as they do in run α. Processes in g_i communicate only with other processes in g_i and F_i. We can use A to solve $\mathcal{SC}(k, t, \text{RV}2)$, and by the validity RV2, we have that all correct processes, and in particular those in g_i, decide v_i. Processes in g_i cannot distinguish run α and run α_i. Hence, since they decide v_i in α_i, they have to decide v_i also in α. It follows that $k + 1$ values are decided in α. □

3.2.2 Protocols

In this section, we provide protocols for the MP/Byz model. We start by observing that PROTOCOL A, used for the crash model, solves $\mathcal{SC}(\text{WV}2)$ also in the MP/Byz model, though only for a restricted range of values of k and t.

Lemma 3.12. *PROTOCOL A solves $\mathcal{SC}(k, t, \text{WV}2)$ in the MP/Byz model for $t < n/2$ and $k \geq \frac{n-t}{n-2t} + 1$.*

Proof. We start by proving termination. Since there are at most t failures, correct processes are guaranteed to receive at least $n - t$ messages and thus they decide.

Next we prove agreement. To have a bound on the number of possible decisions, we look at how many values different from the default value can be decided. Let f be the number of actual failures. We have that any group of $n - t - f$ correct processes that start with the same value can be forced by the f faulty processes to decide that value. Notice that since $f \leq t < n/2$ we have that $n - t - f \geq 1$.

Hence, the number of decisions can be as big as the number of possible disjoint groups of $n - t - f$ correct processes, plus one to take into account the default value. There can be at most $(n - f)/(n - t - f)$ such groups. This function is an increasing function of f, and thus it achieves its maximum value for $f = t$. Hence, the number of different decisions we can have is at most $(n - t)/(n - 2t) + 1$. Since $k \geq (n - t)/(n - 2t) + 1$, agreement is satisfied.

Finally, we prove validity. Assume that all processes are correct and start with v. Then, clearly v is the only decision. □

Lemma 3.13. PROTOCOL A *solves $\mathcal{SC}(k, t, \text{WV}2)$ in the MP/Byz model for $t \geq n/2$ and $k \geq t + 1$.*

Proof. Termination and validity are as in the previous lemma. Next, we prove agreement. Let f be the number of actual failures. We distinguish two cases: 1) $f \leq n - t - 2$ and 2) $f > n - t - 2$. In case 1, we have that for any $n - t$ messages received by a process, at least two of them are sent by correct processes. Hence, for each different value $v \neq v_0$ decided by some process, at least two correct processes have sent that value. Hence, no more than $n/2$ values different from the default value v_0 can be decided. Hence, at most $n/2 + 1$ different values can be decided in case 1. In case 2, the number of correct processes, and thus the number of different decisions by correct processes, is strictly less than $t + 2$.

Putting together the two cases, we can then conclude that the number of different decisions is at most $\max\{n/2 + 1,\ t + 1\} = t + 1 \leq k$. □

Next, we provide a generalized version of the "echo" protocol of Bracha and Toueg [11], which we call ℓ-echo, where $\ell \geq 2$. (The 1-echo protocol is Bracha and Toueg's echo protocol.) The ℓ-echo protocols will be used to provide a family of protocols for $\mathcal{SC}(\text{SV}2)$.

ℓ-`echo protocol`. To ℓ-echo broadcast a message m, the sender s sends the message $\langle \texttt{init}, s, m \rangle$ to all other processes. When a process p receives the first $\langle \texttt{init}, s, m \rangle$ from s, it sends the message $\langle \texttt{echo}, s, m \rangle$ to all other processes. Subsequent $\langle \texttt{echo}, s, m \rangle$ messages from s are ignored. If process p receives message $\langle \texttt{echo}, s, m \rangle$ from more than $(n + \ell t)/(\ell + 1)$ processes, then process p *accepts* message m as sent by the sender process s.

Lemma 3.14. *In a system with $t < \ell n/(2\ell + 1)$, if a sender s uses the ℓ-echo protocol to send a message m then:*

1. *Correct processes accept at most ℓ different messages.*
2. *If s is correct, every correct process accepts m.*

Proof. First we prove case 1. By sake of contradiction, assume that correct processes accept $\ell+1$ different messages $m_1, m_2, \dots, m_{\ell+1}$. Then there must be $\ell+1$ correct processes, say $p_1, p_2, \dots, p_{\ell+1}$, such that process p_i receives more than $(n+\ell t)/(\ell+1)$ echos with m_i, for each $i = 1, 2, \dots, \ell+1$. Thus there must be a total of more than $n+\ell t$ echos sent for the messages $m_1, m_2, \dots, m_{\ell+1}$. Let f be the actual number of faulty processes. Since a faulty process can send $\ell+1$ different echos (it can echo m_1 to p_1, m_2 to p_2, etc.) we have that strictly more than $n+\ell t-(\ell+1)f \geq n+\ell f-(\ell+1)f = n-f$ echos are sent by correct processes. This implies that at least one correct process sent two different echos, which is not possible.

Now we prove case 2. If the sender is correct, then it sends an `init` message for m to all other processes. Any correct process will receive this and broadcast an echo message for m.

It is easy to verify that $t \leq (n+\ell t)/(\ell+1)$; indeed assuming $t > (n+\ell t)/(\ell+1)$ implies $t > n$, which is impossible. Since there are at most t faulty processes and $t \leq (n+\ell t)/(\ell+1)$, no correct process accepts any message other than m. Since there are at least $n-t$ correct processes, it is sufficient that $n-t$ be strictly greater than $(n+\ell t)/(\ell+1)$ in order to guarantee that any correct process receives enough echo messages to be able to accept m. Since $t < \ell n/(2\ell+1)$, we have that $n-t > (n+\ell t)/(\ell+1)$. □

The ℓ-echo protocol is used to define a family of protocols for $\mathcal{SC}(k, t, \mathrm{sv}2)$ as follows:

PROTOCOL C(ℓ). Each process broadcasts its input using the ℓ-echo protocol and waits for $n-t$ messages to be accepted, where one of these $n-t$ messages is the process' own message. If $n-2t$ messages contain the same value v, then the process decides v, else it decides a default value v_0.

Lemma 3.15. PROTOCOL C(ℓ) *solves* $\mathcal{SC}(k, t, \mathrm{sv}2)$ *in the MP/Byz model for* $t < \frac{k-1}{2k+\ell-1}n$ *and* $t < \frac{\ell}{2\ell+1}n$.

Proof. We start by proving termination. Since there are at least $n-t$ correct processes, by Lemma 3.14, each correct process eventually accepts at least $n-t$ messages broadcast by ℓ-echo and is able to make a decision.

Now we prove agreement. For a contradiction, assume that $k+1$ values are decided. One of them could be the default value, but at least k values, different from the default value, are decided. Let $v_1, \dots, v_k$ be these values. To have a correct process p_j decide value v_i, it is necessary (see protocol PROTOCOL C(ℓ)) that there be a set g_i of at least $n-2t$ processes, such that process p_j accepts a value v_i from each process in g_i. Hence, the overall number of values accepted by correct processes with the ℓ-echo protocol is at least $k(n-2t)$. Each faulty process can send ℓ of these values. Since there are at most t faulty processes, the number of different senders is at least $k(n-2t)-(\ell-1)t$. However, since $t < \frac{k-1}{2k+\ell-1}n$, we have that $k > \frac{n+(\ell-1)t}{n-2t}$, and thus $k(n-2t)-(\ell-1)t > n$, which implies that there must be more than n processes, a contradiction.

Finally, we prove validity. Assume that all correct processes start with value v. We have to prove that a correct process decides v.

Let p be a correct process. First, we observe that since p starts with v, it either decides v or v_0. Hence, it suffices to prove that p receives at least $n-2t$ messages with v. Among the $n-t$ messages p receives, at least $n-2t$ are from correct processes. Hence, process p receives at least $n-2t$ messages with v. □

Finally, we provide a protocol for $\mathcal{SC}(\mathrm{wv})1$.

PROTOCOL D. Processes $p_1, p_2, \dots, p_{t+1}$ each broadcasts its input value. A process that receives a value v_i from p_i, $i \in \{1, 2, \dots, t+1\}$, broadcasts an $\langle \texttt{echo}, v_i, p_i \rangle$ message and never echos a value for p_i again. Each process $p_1, p_2, \dots, p_k$ decides on its own value. Every other process decides the first value v_i, $i \in \{1, \dots, t+1\}$, for which it receives identical $\langle \texttt{echo}, v_i, p_i \rangle$ from $n-t$ processes.

In PROTOCOL D, we say that a process *accepts* a value v_i from p_i if it receives identical echos for v_i from at least $n-t$ processes. We define the following functions:

$$V(n,t,f) = \begin{cases} n-f & \textit{if } n-t-f \leq 0 \\ t+1-f+f\lfloor \frac{n-f}{n-t-f} \rfloor & \textit{if } n-t-f > 0 \end{cases}$$

and

$$Z(n,t) = \max_{0 \leq f \leq t} \{\min\{V(n,t,f), n-f\}\}.$$

Lemma 3.16. PROTOCOL D *solves* $\mathcal{SC}(k, t, \mathrm{wv}1)$ *in the MP/Byz model for* $k \geq Z(n,t)$.

Proof. We start by proving termination. At least one process among $p_1, \dots, p_{t+1}$ is correct, and at least $n-t$ receive its value and echo it. Hence, it is guaranteed that each correct process receives at least one set of identical $n-t$ echo messages, and thus is able to decide.

Next, we prove validity. Assume that there are no failures. Then all processes are correct, and thus the values accepted by any process are input values. All decisions are one of the accepted values. Hence, validity WV1 is satisfied.

Finally, we prove agreement. We compute an upper bound on the number of different decisions for each possible value of f; that is, the number of actual failures. By definition, $0 \leq f \leq t$. We distinguish two cases: 1) $n-t-f \leq 0$ and 2) $n-t-f > 0$. In case 1, a correct process may be forced to communicate only with faulty processes. In this case we simply bound the number of decisions with the number of correct processes, that is $n-f$. In case 2, the total number of values that correct processes accept from one faulty process is bounded by $\lfloor \frac{n-f}{n-t-f} \rfloor$. Indeed, a correct process accepts a value when receiving at least $n-t$ echos, at least $n-t-f$ of which are from correct processes. Thus, the total number of values from $p_1, \dots, p_{t+1}$ accepted by correct processes is at most $(t+1-f)+f\lfloor \frac{n-f}{n-t-f} \rfloor$; that is, the number of values sent by correct processes plus the number of values that correct processes may be forced to accept because of the Byzantine behavior of faulty processes. Hence, the number of different decisions that we can have is $t + 1 - f + f\lfloor\frac{n-f}{n-t-f}\rfloor$. It is possible that this bound is bigger than $n - f$. In such a case, we can bound the number of different decisions by $n - f$. Summarizing the two cases, we have that for any f, we bound the number of decisions by $n - f$ if $n - t - f \leq 0$, and by $\min\{t + 1 - f + f\lfloor\frac{n-f}{n-t-f}\rfloor, n - f\}$ if $n - t - f > 0$. The maximum overall possible values of f is given by $Z(n, t)$. Hence, we have that the number of decisions is always at most $Z(n, t)$, as required. □

We note that when $t < \frac{n}{3}$, $\lfloor\frac{n-f}{n-t-f}\rfloor = 1$ for all $0 \leq f \leq t$, and therefore, the protocol above guarantees agreement for any $k > t$ (see Fig. 4).

4 Shared Memory Models

In this section, we consider $\mathcal{SC}$ in the shared memory model. We assume that processes communicate by means of a shared memory that provides single-writer multireader atomic registers [22]. Here, "single-writer" means that there is a single designated process that is allowed write to the variable; any other process—even if Byzantine faulty—is prohibited from writing to it. An atomic register provides read and write operations that appear to occur sequentially, i.e., the views of individual processes conform with some sequential history of all performed operations [22].

The shared memory does not fail, though processes accessing it may. We note that this model is motivated by many recent middleware systems that provide shared memory emulation using replication to mask the arbitrary (Byzantine) failure of processes implementing these abstractions. These middleware systems generally guarantee that shared objects themselves do not "fail," and hence, that their integrity, safety properties, and access interfaces and restrictions are preserved. Nevertheless, since legitimate clients accessing these objects might fail arbitrarily, they could corrupt the states of these objects in any way allowed by the object interfaces.

We assume that the system is asynchronous; that is, processes may take an arbitrary (but finite) time to execute a step, and reads and writes may take an arbitrary (but finite) time to complete.

The following simple transformation simulates any protocol X for the message passing model using a shared memory protocol (for both crash failures and Byzantine failures):

SIMULATION. Whenever protocol X prescribes that p send its ith message m to process q, p writes m to a single-writer single-reader register designated for p's ith message to q; q repeatedly reads the register until it reads a value there. Similarly, when protocol X prescribes that p send its ith broadcast m, p writes m to a single-writer multi-reader register designated for p's ith broadcast; each process repeatedly reads the register until it reads a value there.

Hence, every algorithm for MP/CR (similarly, MP/Byz) works for the SM/CR (SM/Byz) model. In many cases, however, direct algorithms for the shared memory model cover wider areas of possibility, as detailed below.

Recall that the impossibility result of [9], [20], [30] (i.e., Lemma 3.2) also applies to the shared memory model (SM/CR).

4.1 Shared Memory Model with Crash Failures

In this section, we consider the shared-memory crash (SM/CR) model. In Sections 4.1.1 and Section 4.1.2, we provide impossibility results and protocols, respectively. Fig. 5 shows a graphical representation of the results.

Our impossibility results and protocols for the SM/CR model leave a small gap for $\mathcal{SC}(\text{SV}2)$.

4.1.1 Impossibilities

Lemma 4.1. *In the SM/CR model, there is no protocol for $\mathcal{SC}(k, t, \text{WV}1)$ for $k \leq t$.*

Proof. (Similar to Lemma 3.4.) For a contradiction, assume that there exists such a protocol A. We claim that A can be used to solve $\mathcal{SC}(k, t, \text{RV}1)$ for $t \geq k$. To see why, consider any run α in which $f \leq t$ processes are faulty. Let g be the set of correct processes in α and g' be the set of faulty processes.

Now consider a run α' that is identical to α except that all processes are correct and for each process p_j in g', any invocation for a write operation after the time p_j failed in α is delayed until after all processes of g decide. That is, for each $p_i \in g$, α and α' are indistinguishable up to the time when all processes in g decide in α (and thus in α'). By the validity condition WV1, each process decides on some process' input in α'. Since processes in g cannot distinguish between α and α', they must decide the same value in α as they decide in α', and so validity RV1 is satisfied in α. In other words, protocol A solves $\mathcal{SC}(k, t, \text{RV}1)$ for $t \geq k$, contradicting Lemma 3.2. □

Lemma 4.2. *In the SM/CR model, there is no protocol for $\mathcal{SC}(k, t, \text{SV}1)$.*

Proof. (Similar to Lemma 3.5.) For a contradiction, assume that there exists such a protocol A. Let α be an execution of A in which all processes are correct and they all start with different values. Let v a decision made by at least two processes (there is always such a decision since $k < n$). Because of validity SV1, v is the input of some process p_i and since all inputs are different only p_i has v as input.

Use A in the run α' that is the same as α except that process p_i crashes right after completing its last write operation.

Clearly, α and α' are indistinguishable and thus each correct process makes the same decision in both runs. Hence, in α' value v is decided by at least one process p_j, $j \neq i$. But only p_i has v as input and p_i is not correct in α', and so validity SV1 is violated. □

Lemma 4.3. *In the SM/CR model, there is no protocol for $\mathcal{SC}(k, t, \text{SV}2)$ when $t \geq \frac{n}{2}$ and $t \geq k$.*

Proof. By sake of contradiction, assume that such a protocol A exists. Use A in a run α constructed as

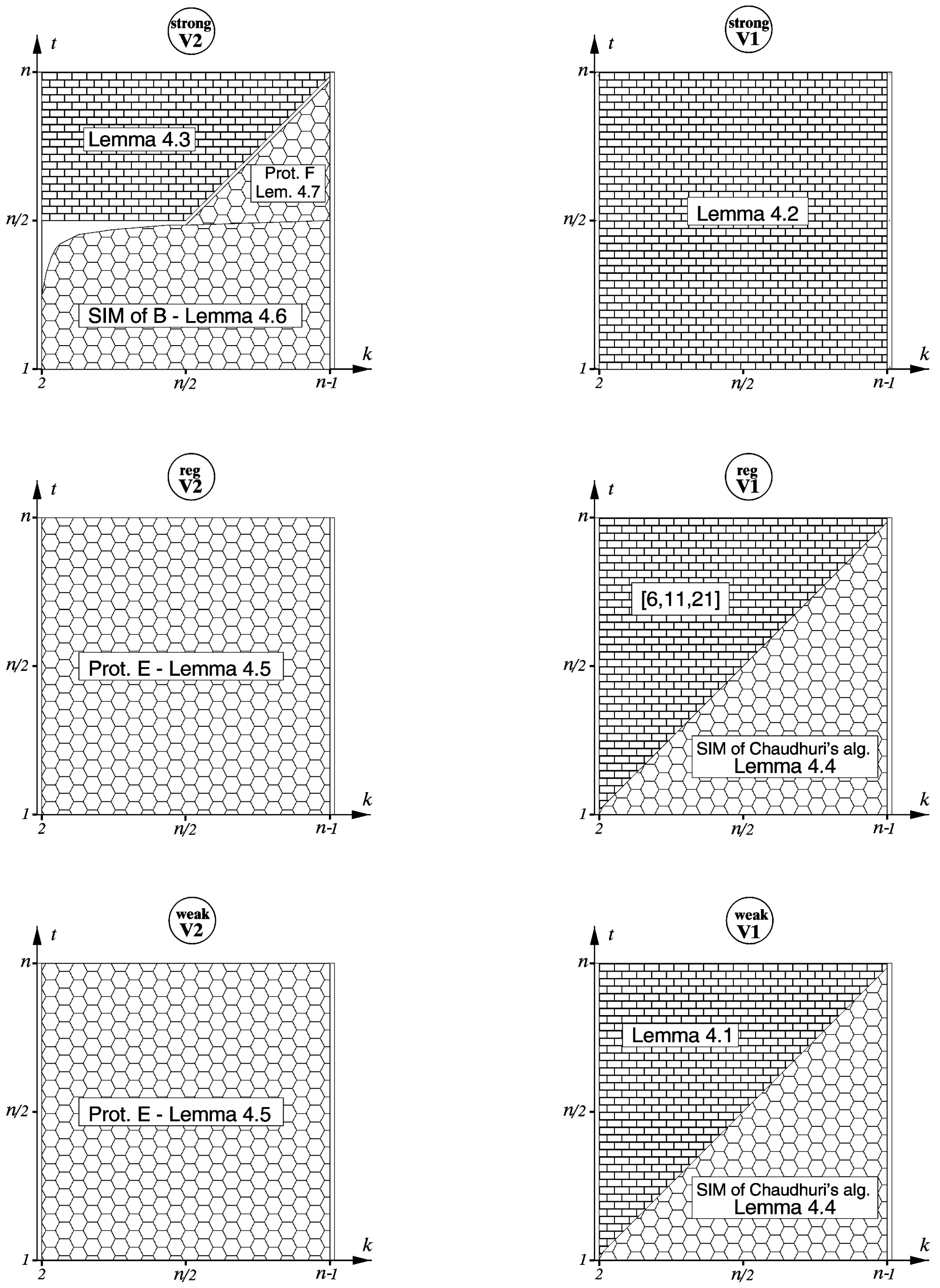

Fig. 5. Shared memory crash model (SM/CR). Regions filled in brick pattern indicate impossibilitiy. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn for the case $n = 64$ processes.

follows: Processes $p_1, p_2, \ldots, p_{t+1}$ each start with a different value, say $v_1, v_2, \ldots, v_{t+1}$. Let g be the set of these processes and g' be the remaining processes. Processes in g' do not take any step until after all processes in g decide. Now, for each $i = 1, 2, \ldots, t+1$, consider a run α_i constructed as follows: Processes in g start with the input as in α and processes in g' all start with v_i. Processes in g, except process p_i, fail after p_i decides. Note that since $t \geq n/2$, we have that $\{p_i\} \cup g'$ has at least $n - t$ processes, and hence a decision by p_i must eventually be reached.

Use A to solve the problem in α_i. By validity SV2, since all correct processes start with v_i, all have to decide v_i, and in particular process p_i decides v_i.

Since α_i and α are indistinguishable for p_i, p_i has to decide v_i also in α. This is true for each $i = 1, 2, \ldots, t+1$. Hence, at least $t+1$ different values are decided in α. But $t + 1 > k$. This contradicts the fact that A solves the problem. □

4.1.2 Protocols

In this section, we sketch protocols for the SM/CR model. As mentioned above, since the SM/CR model is "more powerful" than the MP/CR model any protocol for the MP/CR model can be transformed into a protocol for the SM/CR model using SIMULATION (see also chapter 17 of [25]). Hence, all of the protocols we have seen in Section 3.1.2 can be used here (after the transformation).

Lemma 4.4. SIMULATION *of Chaudhuri's protocol [13] solves* $\mathcal{SC}(k,t,\text{RV}1)$ *in the SM/CR model for* $t<k$.

The solvability region of $\mathcal{SC}(\text{RV}1)$, as well as that of $\mathcal{SC}(\text{WV}1)$, is as for the message passing model. For other validity conditions we can do better. Next, we give a protocol for $\mathcal{SC}(\text{RV}2)$.

PROTOCOL E. Each process writes its own input into a single-writer register. The process then scans the registers of all other processes exactly once. If all the values it reads in this single scan (including its own) are identical, it decides that value, otherwise, it decides v_0 (a default value).

Lemma 4.5. PROTOCOL E *solves* $\mathcal{SC}(k,t,\text{RV}2)$ *in the SM/CR model for* $k\geq 2$.

Proof. Termination is trivial. Next, we prove agreement. Let v be the value written in the first write to complete. Every process reads v in its scan and decides either v or v_0. Therefore, at most two values are decided. Next, we prove validity. If all of the processes start with the same value v, then this is the only value written and so the only possible decision value. □

The next two lemmas consider $\mathcal{SC}(\text{SV}2)$.

Lemma 4.6. *SIMULATION of* PROTOCOL B *solves* $\mathcal{SC}(k,t,\text{SV}2)$ *in the SM/CR model for* $t<\frac{k-1}{2k}n$.

PROTOCOL F. Each process writes its own input into a single-writer register. The process then scans the registers of all other processes repeatedly, until in a single scan of all registers it successfully reads from some $r\geq n-t$ process' registers. If $r\leq t$ (possible if $n\leq 2t$), then the process decides on its own input. Otherwise, i.e., if $r=t+i$ for some $i\geq 1$, then it decides its own input if at least i registers of these r (including its own) hold its input value, and a default value v_0 otherwise.

Lemma 4.7. PROTOCOL F *solves* $\mathcal{SC}(k,t,\text{SV}2)$ *in the SM/CR model for all* $k>t+1$.

Proof. We first prove termination. Every correct process eventually writes its register. Thus, every correct process will eventually scan the registers reading at least $n-t$ values. Next we prove agreement. A process can decide its own value $v\neq v_0$, either if it reads $r\leq t$ values or if it reads $r=t+i$ values and at least i values are equal to v.

Since a process writes before scanning the registers, we have that as long as less than $t+1$ writes have been completed, less than $t+1$ values have been decided upon, and each of these values is one of the values written. Fix the point in the execution where $t+1$ writes have been completed and let $v_1,v_2,\ldots,v_{t+1}$ be the values written. From that point on, any decided value, different from the default value, must be one of $v_1,\ldots,v_{t+1}$. Indeed, from that point on, any process that scans the memory will read $r=t+i$ values, with $i\geq 1$; and since, in order to decide a value v, at least i of them must be equal to v, it must be that v is one of $v_1,\ldots,v_{t+1}$ (there are only $i-1$ other values). Hence, considering also the default value, we have that at most $t+2$ values are decided.

Finally, we prove validity. If all the correct processes start with the same input value v, then whenever $t+1$ or more values are read by a correct process, at most t of them differ from v and therefore the process decides v.□

4.2 Shared Memory Model with Byzantine Failures

In this section, we consider the SM/Byz model where processes may exhibit Byzantine behavior. In Section 4.2.1 and Section 4.2.2, we provide impossibility results and protocols, respectively. Fig. 6 shows a graphical representation of the results provided in this section.

Our impossibility results and protocols leave a substantial gap for $\mathcal{SC}(\text{WV}1)$, and very small gaps for $\mathcal{SC}(\text{SV}2)$ and $\mathcal{SC}(\text{RV}2)$.

4.2.1 Impossibilities

In this section, we provide impossibility results for the SM/Byz model. Clearly, the impossibilities proven for the SM/CR model still hold. In particular, the impossibilities for $\mathcal{SC}(\text{WV}1)$, $\mathcal{SC}(\text{SV}2)$, and $\mathcal{SC}(\text{SV}1)$ are directly derived from the corresponding ones for the SM/CR model. Next, we provide additional impossibilities.

Lemma 4.8. *In the SM/Byz model, there is no protocol for* $\mathcal{SC}(k,t,\text{RV}1)$.

Proof. The proof is the same as the one of Lemma 3.10 (that proof does not rely on the fact that the system is MP). □

Lemma 4.9. *In the SM/Byz model, there is no protocol for* $\mathcal{SC}(k,t,\text{RV}2)$ *for* $t\geq\frac{n}{2}$ *and* $t\geq k$.

Proof. (Similar to Lemma 4.3.) By sake of contradiction, assume that such an algorithm A exists. Use A in a run α constructed as follows. Processes $p_1,p_2,\ldots,p_{t+1}$ each start with a different value, say $v_1,v_2,\ldots,v_{t+1}$. Let g be the set of these processes and g' be the remaining processes. Processes in g' do not take any step until after all processes in g decide (since $t\geq n-t$, processes in g must decide).

Now consider for each $i=1,2,\ldots,t+1$ a run α_i constructed as follows: All processes start with v_i but processes in g, except p_i, are faulty and $p_j\in g$, for each $j\neq i$, claims to have v_j as input. Processes in g' do not take any step until after p_i decides. Clearly, runs α_i and α are indistinguishable for p_i.

Use A to solve the problem in α_i. Since $t\geq n/2$, p_i must decide even if communicating only with faulty processes. By validity RV2, since all processes start with v_i, all correct have to decide v_i, and in particular, process p_i decides v_i.

Since α_i and α are indistinguishable for p_i, p_i has to decide v_i also in α. This is true for each $i=1,2,\ldots,t+1$. Hence, at least $t+1$ different values are decided in α. But $t+1>k$. Hence, A does not solve the problem and there is a contradiction. □

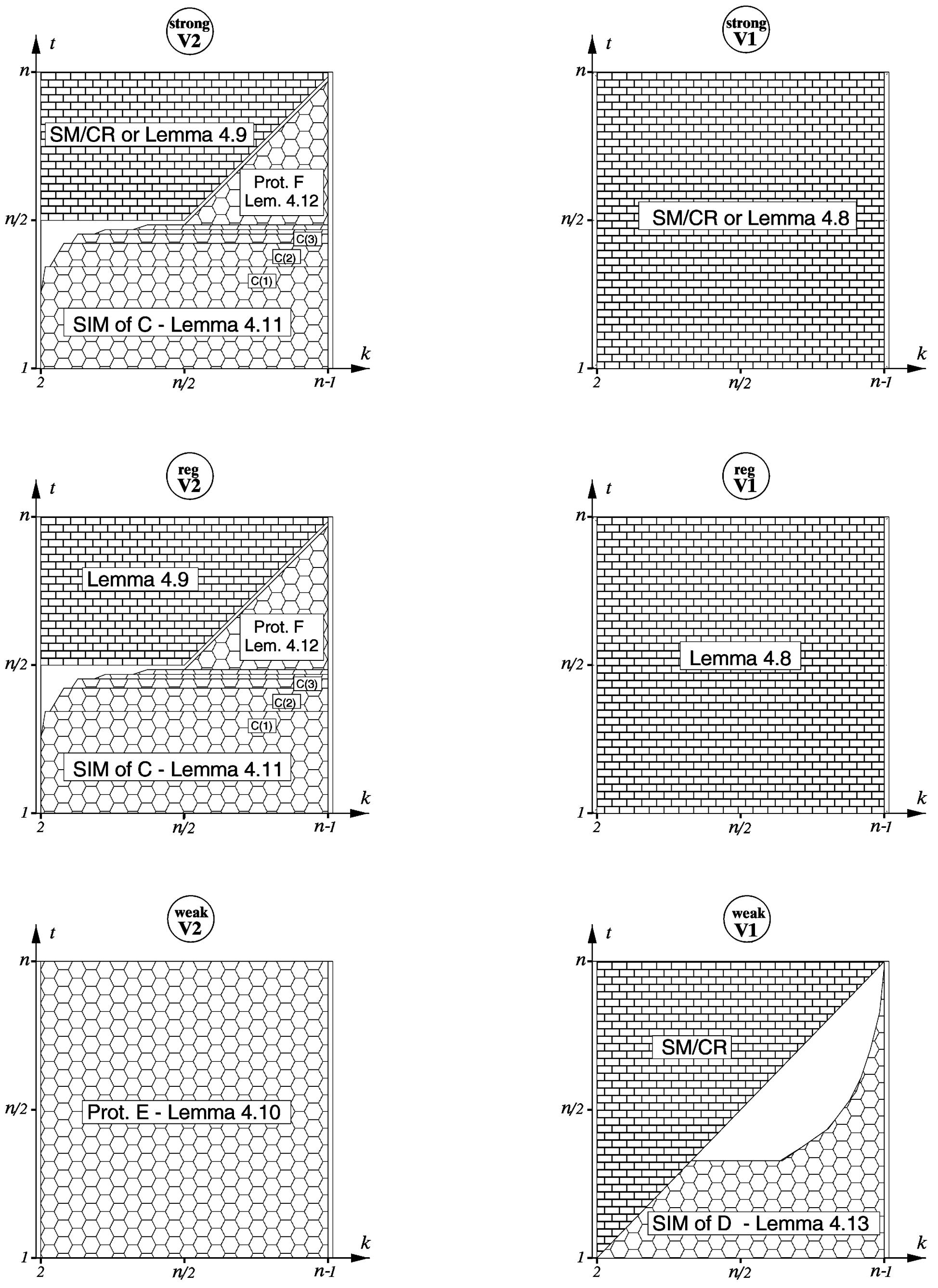

Fig. 6. Shared-memory Byzantine model (SM/Byz). Regions filled in brick pattern indicate impossibility. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn for the case $n = 64$ processes.

4.2.2 Protocols

In this section, we sketch protocols for the SM/Byz model. Here again we use the SIMULATION to transform some of our protocols for the MP/Byz to the SM/Byz model.

Lemma 4.10. PROTOCOL E *solves* $\mathcal{SC}(k, t, \text{wv}2$ *for* $k \geq 2$ *in the SM/Byz model.*

Proof. Termination is trivial. Next, we prove agreement. Let v be the value written in the first write by a correct process to complete. Every correct process reads v in its scan (because a process writes before reading and v is the first write by a correct process to complete) and decides either v or v_0. Therefore, at most two values are decided.

Next, we prove validity. If all of the processes start with the same value v and all are correct, then v is the only value written and so the only possible decision value. □

The next two lemmas consider $\mathcal{SC}(\text{sv}2)$.

Lemma 4.11. SIMULATION *of* PROTOCOL C*(ℓ) solves* $\mathcal{SC}(k, t, \text{sv}2)$ *in the SM/Byz model for* $t < \frac{k-1}{2k+\ell-1}n$ *and* $t < \frac{\ell}{2\ell+1}n$.

Lemma 4.12. PROTOCOL F *solves* $\mathcal{SC}(k, t, \text{sv}2)$ *in the SM/Byz model for* $k > t + 1$.

Proof. (Similar to Lemma 4.7.) We first prove termination. Every correct process eventually writes its register. Thus, every correct process will eventually scan the registers, reading at least $n - t$ values.

Next, we prove agreement. A process can decide its own value $v \neq v_0$, either if it reads $r \leq t$ values in a scan or if it reads $r = t + i$ values in a scan and at least i values are equal to v. Since a process writes before scanning the registers, we have that as long as less than $t + 1$ writes by *correct* processes have been completed, less than $t + 1$ values other than the default have been decided upon by *correct* processes, and each of these values is one of the values written by correct processes. Fix the point in the execution where $t + 1$ writes by correct processes have been completed and let $v_1, v_2, \ldots, v_{t+1}$ be the values written. From that point on, any value decided by a correct process, except the default value, must be one of $v_1, \ldots, v_{t+1}$. Indeed, from that point on, any process that scans the memory will read $r = t + i$ values with $i \geq 1$, and since in order to decide a value v at least i of them must be equal to v, it must be that v is one of $v_1, \ldots, v_{t+1}$ (there are only $i - 1$ other values). Hence, considering also the default value, we have that at most $t + 2$ values are decided.

Finally, we prove validity. If all the correct processes start with the same input value v, then whenever $t + 1$ or more values are read by a correct process, at most t of them differ from v, and therefore the process decides v. □

Finally, we also can use PROTOCOL D in the SM/Byz by means of the SIMULATION. Recall that $Z(n, t)$ has been defined before Lemma 3.16.

Lemma 4.13. *SIMULATION of PROTOCOL D solves* $\mathcal{SC}(k, t, \mathrm{WV1})$ *in the SM/Byz model for* $k \geq Z(n, t)$.

5 Conclusions

We have considered several variations of the *k*-set consensus problem. The variations were obtained by considering six different validity conditions. One of these variations is the *k*-set consensus problem introduced by Chaudhuri and considered by several papers in the literature (the other variations have been considered for the classical, i.e., $k = 1$, consensus problem). We showed that the exact definition of the validity condition is crucial in order to discern solvable from impossible. Known results have demarcated this line for the problem considered by Chaudhuri. In this paper, we have provided this line for the other variations of the problem. The results show that this line changes depending on the exact definition of the validity condition. We have considered each of the variations in the message-passing and the shared memory models and for each of these models we considered crash and Byzantine failures. In most of the cases, we were able to exactly demarcate the line between solvable and impossible; in a few cases there is still a gap to be filled.

In most of our protocols for the Byzantine failure model, processes are required to "help" other processes by continually participating in the (echo) protocol. Therefore, termination is satisfied only in the sense that correct processes decide, but not in the sense that they are guaranteed to eventually stop. It is currently open whether there exists terminating protocols for the same settings.

Acknowledgments

Portions of Sections 1, 2, and 4 appeared previously in a preliminary version (see [14]).

References

[1] H. Attiya, "A Direct Proof of the Asynchronous Lower Bound for *k*-Set Consensus," *Proc. 17th ACM Symp. Principles of Distributed Computing,* pp. 314, July 1998.

[2] H. Attiya, A. Bar-Noy, D. Doev, D. Koller, D. Peleg, and R. Reischuk, "Achievable Cases in an Asynchronous Environment," *IEEE Symp. Foundations of Computer Science,* 1987.

[3] H. Attiya, D. Dolev, and J. Gil, "Asynchronous Byzantine Consensus," *Proc. Symp. Principles of Distributed Computing,* 1984.

[4] H. Attiya, A. Bar-Noy, and D. Dolev, "Sharing Memory Robustly in Message-Passing Systems," *J. ACM,* vol. 42, no. 1, pp. 124–142, Jan. 1995.

[5] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk, "Renaming in an Asynchronous Environment," *J. ACM,* vol. 37, no. 3, pp. 524–548, July 1990.

[6] H. Attiya and S. Rajsbaum, "A Combinatorial Framework for Wait-Free Computability," *Proc. 10th Int'l Workshop Distributed Algorithms,* Oct. 1996.

[7] H. Attiya and J. Welch, *Distributed Computing.* McGraw Hill, 1998.

[8] M. Ben-Or, "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols," *Proc. Symp. Principles of Distributed Computing,* 1983.

[9] E. Borowsky and E. Gafni, "Generalized FLP Impossibility Result for *t*-Resilient Asynchronous Computations," *Proc. 25th ACM Symp. Theory of Computing,* pp. 91–100, 1993.

[10] G. Bracha, "An $o(n \log n)$ Expected Rounds Randomized Byzantine Generals Algorithm," *Proc. Fourth ACM Symp. Principles of Distributed Computing (PODC),* 1985.

[11] G. Bracha and S. Toueg, "Resilient Consensus Protocols," *Proc. Second ACM Symp. Principles of Distributed Computing,* pp. 12–26, 1983.

[12] T.D. Chandra, V. Hadzilacos, and S. Toueg, "The Weakest Failure Detector for Solving Consensus," *J. ACM,* vol. 43, no. 4, pp. 685–722, July 1996.

[13] S. Chaudhuri, "More Choices Allow More Faults: Set Consensus Problems in Totally Asynchronous Systems," *Information and Computation,* vol. 105, no. 1, pp. 132–158, July 1993.

[14] R. De Prisco, D. Malkhi, and M. Reiter, "On *k*-Set Consensus Problems in Asynchronous Systems," *Proc. 18th ACM Symp. Principles of Distributed Computing,* pp. 257–265, May 1999.

[15] D. Dolev, C. Dwork, and L. Stockmeyer, "On the Minimal Syncrhony Needed for Distributed Consensus," *J. ACM,* vol. 34, no. 1, pp. 77–97, Jan. 1987.

[16] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the Presence of Partial Synchrony," *J. ACM.* vol. 35, no. 2, pp. 288–323, Apr. 1988.

[17] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM,* vol. 32, no. 2, pp. 374–382, Apr. 1985.

[18] M. Herlihy and S. Rajsbaum, "Algebraic Spans," *Proc. 14th ACM Symp. Principles of Distributed Computing,* pp. 90–99, 1995.

[19] M. Herlihy, S. Rajsbaum, and M. Tuttle, "Unifying Synchronous and Asynchronous Message-Passing Models," *Proc. 17th ACM Symp. Principles of Distributed Computing,* pp. 233–142, 1998.

[20] M. Herlihy and N. Shavit, "The Asynchronous Computability Theorem for *t*-Resilient Tasks," *Proc. 25th ACM Symp. Theory of Computing,* pp. 111–120, 1993.

[21] L. Lamport, "The Weak Byzantine Generals Problem," *J. ACM,* vol. 30, no. 3, pp. 254-280, 1983.

[22] L. Lamport, "On Interprocess Communication, Part II: Algorithms," *Distributed Computing,* vol. 1, pp. 86–101, 1986.

[23] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems,* vol. 4, no. 3, pp. 382–401, July 1982.

[24] M. Loui and H. Abu-Amara, "Memory Requirements for Agreement among Unreliable Asynchronous Processes," *Parallel and Distributed Computing*, pp. 163–183. Greenwich, Conn.: JAI Press, 1987.

[25] N. Lynch, *Distributed Algorithms.* San Francisco: Morgan Kaufmann, 1996.

[26] N. Lynch and S. Rajsbaum, "On the Borowsky-Gafni Simulation Algorithm," *Proc. Fourth Israeli Symp. Theory of Computing and Systems*, pp. 4–15, June 1996.

[27] G. Neiger, "Distributed Consensus Revisited," *Information Processing Letters*, vol. 49, no. 4, pp. 195–201, 1994.

[28] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[29] M. Rabin, "Randomized Byzantine Generals," *Proc. 15th ACM Symp. Theory of Computing (STOC)*, pp. 403–409, 1983.

[30] M. Saks and F. Zaharoglou, "Wait-Free *k*-Set Agreement is Impossible: The Topology of Public Knowledge," *Proc. 25th ACM Symp. Theory of Computing*, pp. 101–110, 1993.

Roberto De Prisco received the Laurea degree from the University of Salerno (Italy) in 1991, the MS degree from the Massachusetts Institute of Technology in 1997, the Dottorato degree from the University of Naples (Italy) in 1998, and the PhD degree from the Massachusetts Institute of Technology in 2000, all in computer science. Currently, he is a member of the Algorithms Group at the University of Salerno, Italy, and a member of the Theory of Distributed Systems Group at the Laboratory for Computer Science, MIT. His research interests focus on distributed systems and algorithms and data structures.

Dahlia Malkhi received the BSc in mathematics and computer science and the MSc and PhD degrees in computer science in 1985, 1988, and 1994, respectively, from the Hebrew University of Jerusalem, Israel. She was a member of the Secure Systems Research Department at AT&T Labs-Research in Florham Park, New Jersey, from 1995 to 1999. She is currently a faculty member of the School of Computer Science and Engineering at the Hebrew University of Jerusalem, Jerusalem, Israel, where she heads the Secure Systems Research Laboratory. Her research interests include all areas of distributed systems and security.

Michael K. Reiter received the BS degree in mathematical sciences from the University of North Carolina at Chapel Hill in 1989, and the MS and PhD degrees in computer science from Cornell University in 1991 and 1993, respectively. From 1993 to 1998, he conducted research on security and fault tolerance in distributed systems at AT&T Labs-Research (formerly AT&T Bell Labs). In 1998, he joined Bell Labs, Lucent Technologies as director of Secure Systems Research, where he is continuing his research program on computer security and fault tolerance.