On k-set Consensus Problems in Asynchronous Systems

Roberto De Prisco* Dahlia Malkhi† Michael K. Reiter‡

Abstract

In this paper we investigate the k-set consensus problem in asynchronous, message-passing 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 different 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 for 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.

1 Introduction

The k-set consensus problem is an abstraction of many coordination problems in a distributed system that can suffer process failures. Each process begins with an input value and must irrevocably decide on one output value, so that a total of at most k values are decided by correct processes. The set of allowed decision values are specified by a *validity* condition that constrains the decisions of correct processes as a function of the input values and whether failures occur during the run of the protocol.

In this paper we explore the solvability of the k-set consensus problem in asynchronous message passing systems, in models in which processes fail by crashing or fail arbitrarily (Byzantine failures). The main theme in this paper is that the validity condition has a profound impact on when the problem is solvable. We consider six different validity conditions and use these conditions to demarcate when k-set consensus is solvable for each system model. 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 one case we leave a substantial gap.

The k-set consensus problem was introduced by Chaudhuri [5], who studied the problem in asynchronous message passing systems in which processes fail by crashing. The validity condition adopted in [5] requires that each correct process decide on a value that is equal to some input value. In this context, Chaudhuri provided a protocol to solve the k-set consensus problem that tolerates up to $t < k$ process failures. The condition $t < k$ was later proved to be necessary in this context [3, 7, 13] (see also [1]). The six validity conditions we study include the one originally used in [5] and others studied for $k = 1$ (consensus) with crash and Byzantine failures (e.g., see [11]). For $k = 1$, the condition $t = 0$ is known to be necessary for any nontrivial validity condition [6].

The rest of this abstract is structured as follows. In Section 2 we define the problem. We study the k-set consensus problem for message passing systems with crash-failures in Section 3, and for message passing systems with Byzantine failures in Section 4. Section 5 concludes and outlines directions for future work.

2 The problem

We consider a distributed system consisting of n processes denoted by $p_1, p_2, ..., p_n$. A process that follows its protocol specification throughout an execution is said to be *correct*, and a process that departs from its specification is said to be *faulty*. In the *crash model*, faulty processes are allowed to prematurely halt execution only. In the *Byzantine model*, a faulty process can deviate from its specification arbitrarily. We assume that at most t processes fail, where $t \geq 1$ is a known, positive integer.

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. Communication is reliable and authenticated, in the sense that a correct process p_i receives a message m from a correct process p_j if and only if p_j sent m to p_i. Processes may take an arbitrary (but finite) time to execute a step and messages may incur an arbitrary (but finite) delay on the communication network. That is, the system is *asynchronous*.

We denote a k-set consensus problem by $\mathcal{SC}(k)$ or simply $\mathcal{SC}$ when k is not relevant. For any k, $1 \leq k \leq n$, the $\mathcal{SC}(k)$ problem is defined as follows. Each process p_i starts the computation with an input value v_i. 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.

*MIT Lab for Computer Science, 545 Technology Sq. NE43-368, Cambridge MA, 02139, and Dip. di Informatica ed Applicazioni, Università di Salerno, Italy. Email: robdep@theory.lcs.mit.edu. Work done while at AT&T Labs—Research.

†AT&T Labs—Research, 180 Park Ave., Florham Park, NJ 07932. Email: dalia@research.att.com.

‡Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. Email: reiter@research.bell-labs.com.

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.
- 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 any protocol for solving $\mathcal{SC}(D)$ can be used to solve $\mathcal{SC}(C)$ in the same model. Clearly $\mathcal{SC}(C)$ is weaker than $\mathcal{SC}(D)$ if 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 $\mathcal{SC}(C)$. Figure 1 shows the "weaker than" relation among the six validity conditions considered in this paper.

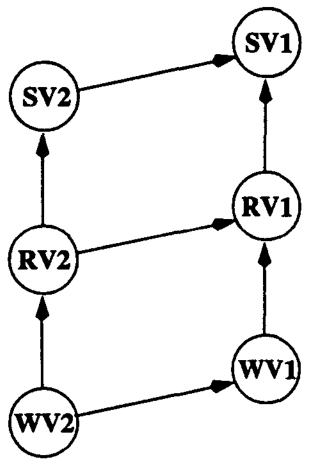

Figure 1: Validity conditions. An arrow from a validity condition C to a validity condition D means that $\mathcal{SC}(C)$ is weaker than $\mathcal{SC}(D)$ (and that $\mathcal{SC}(D)$ is stronger than $\mathcal{SC}(C)$).

$\mathcal{SC}(k,\text{RV}1)$ is the consensus problem as considered by Chaudhuri [5]. $\mathcal{SC}(1,\text{RV}1)$ and $\mathcal{SC}(1,\text{RV}2)$ are classical consensus problems (see, e.g., [10, Ch. 6]). $\mathcal{SC}(1,\text{SV}2)$ has been considered in the Byzantine setting [9, 12]. $\mathcal{SC}(1,\text{WV}2)$ is weak Byzantine agreement [8].

It is well known that $\mathcal{SC}(1)$ cannot be solved for any nontrivial validity condition [6] and, in particular, for any of the validity conditions that we consider here. On the other hand, $\mathcal{SC}(n)$ 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 $2 \le k \le n-1$. Since the problem is easily solvable for $t = 0$ we also assume that $t \ge 1$.

3 Crash failures

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

Lemma 3.1 ([5]) *In the crash model, there is a protocol for $\mathcal{SC}(k,t,\text{RV}1)$, for $t < k$.*

Lemma 3.2 ([3, 7, 13]) *In the crash model, there is no protocol for $\mathcal{SC}(k,t,\text{RV}1)$, for $t \ge k$.*

By Lemma 3.1, we have that $\mathcal{SC}(k,t,\text{RV}2)$, $\mathcal{SC}(k,t,\text{WV}1)$ and $\mathcal{SC}(k,t,\text{WV}2)$ are solvable for $t < k$ because RV2, WV1 and WV2 are weaker than RV1. By Lemma 3.2, $\mathcal{SC}(k,t,\text{SV}1)$ cannot be solved for $t \ge k$ because SV1 is stronger than RV1.

In Sections 3.1 and 3.2, we provide further impossibility results and protocols, respectively. Figure 2 shows a graphical representation of the results provided in this section.

3.1 Impossibilities

In this section we provide impossibility results for the crash 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 crash model, there is no protocol for $\mathcal{SC}(k,t,\text{WV}2)$, for $t \ge \frac{(k-1)n+1}{k}$.*

Proof: For a contradiction, assume that such a protocol A exists. 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 $\mathcal{SC}_{\mathcal{P}}(k,t,\text{WV}2)$.

Since $t \ge ((k-1)n+1)/k$ implies $n \ge k(n-t)+1$, we can partition the n processes into k groups $g_1, g_2, ..., g_k$ of disjoint processes with $g_1, , ..., g_{k-1}$ containing exactly $n-t$ processes and g_k containing at least $n-t+1$ processes. If $t = n$ we let $g_1, g_2, ..., g_{k-1}$ 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 $\mathcal{SC}_{g_k}(1,1,\text{WV}2)$: 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. However, this contradicts [6], since no such protocol exists. 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 $v_1, ..., v_{k-1}$ be $k-1$ values different from v_k, v_{k+1}.

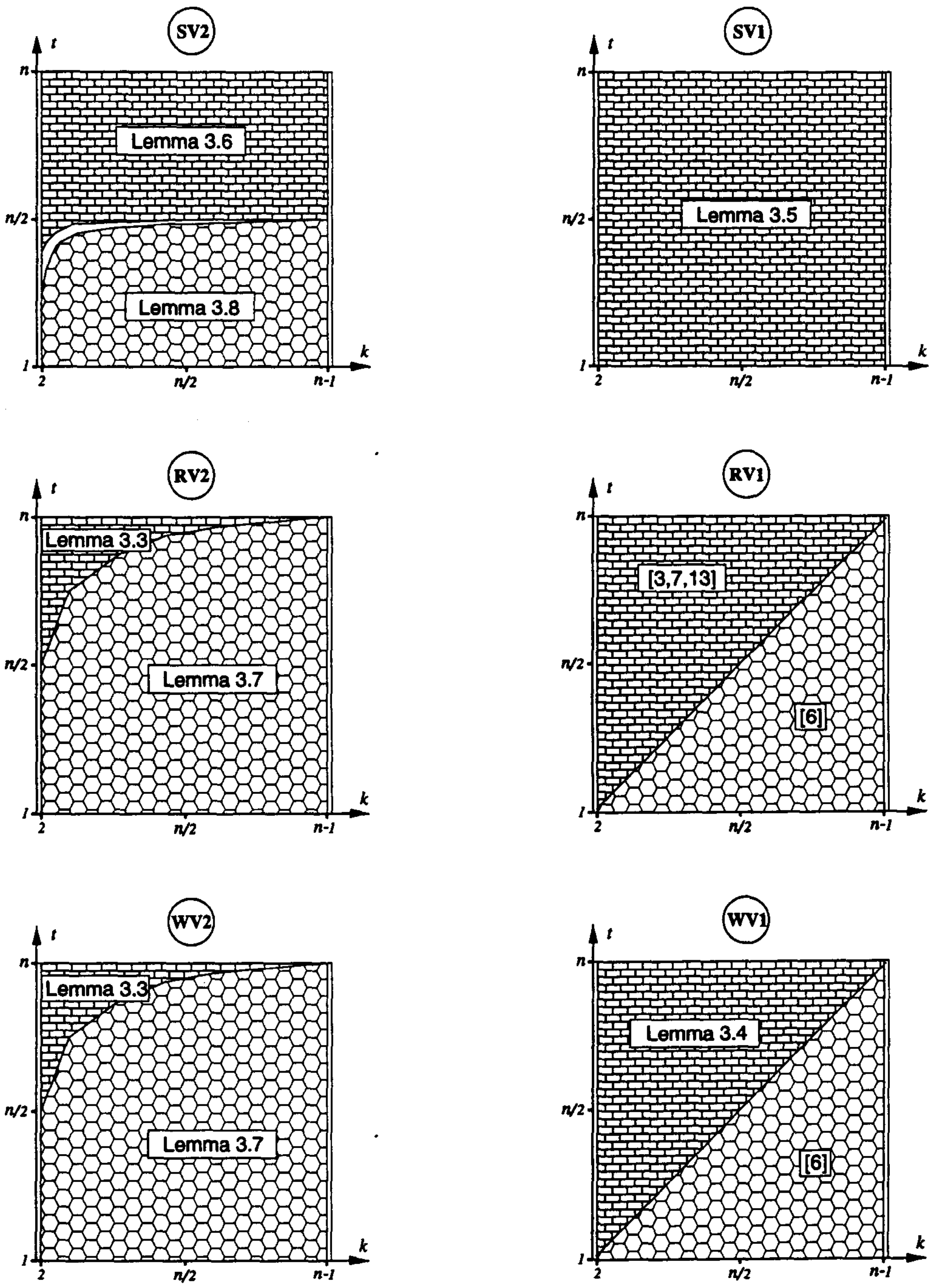

Figure 2: Crash model. Regions filled in brick pattern indicate impossibility. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn to scale $n = 64$.

Fix i, $i \in \{1,2,...,k-1\}$ and consider the following run α_i: all processes are correct, all start with v_i and all messages sent to processes in g_j, $j = 1,2,....,k$ by processes not in g_j are delayed until all processes in g_j make a decision. We can use A to solve $\mathcal{SC}_{\mathcal{P}}(k,t,\text{WV}2)$ and by validity WV2 we have that all processes, in particular those in group g_i, decide v_i.

Now consider the following run α. All processes are correct, for each i, $i = 1,2,...,k-1$, each 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, $i = 1,2,...,k$, 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. We can use A to solve $\mathcal{SC}_{\mathcal{P}}(k,t,\text{WV}2)$ in α. However, for each i, $i = 1,2,...,k$, 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 $i = 1,2,...,k-1$, 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 $\mathcal{SC}_{\mathcal{P}}(k,t,\text{WV}2)$. □

Lemma 3.4 *In the crash model, there is no protocol for $\mathcal{SC}(k,t,\text{WV}1)$, 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 $\mathcal{SC}(k,t,\text{RV}1)$ 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 $\mathcal{SC}(k,t,\text{RV}1)$ for $t \geq k$, contradicting Lemma 3.2. □

Lemma 3.5 *In the crash model, there is no protocol for $\mathcal{SC}(k,t,\text{SV}1)$.*

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 consider 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 crash model, there is no protocol for $\mathcal{SC}(k,t,\text{SV}2)$, 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 $v_1, v_2, ..., v_n$, 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 the following run α_i. 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 $g_1, ..., g_{\lfloor \frac{n-t}{n-2t} \rfloor}$ 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 $k \leq \frac{n-t}{n-2t} - 1 < \lfloor \frac{n-t}{n-2t} \rfloor$. □

3.2 Protocols

In this section we provide two protocols for the crash 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 crash 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 $g_1, g_2, ..., g_k$, 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 $\mathcal{SC}(k,t,$SV2$)$ in the crash model for $t < \frac{k-1}{2k}n$.*

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 $g_1, g_2, ..., g_k$, 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 processes. Hence process p receives at least $n-2t$ messages with v. □

3.3 Remarks

For $\mathcal{SC}$(RV2) and $\mathcal{SC}$(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. Since for all other points on this line the problem is not solvable it would be very surprising if for those isolated points the problem is solvable. For $\mathcal{SC}$(SV2) there is also small gap between our possibility and impossibility results (Lemmas 3.6 and 3.8).

4 Byzantine failures

In this section we consider the Byzantine model. In Section 4.1 we are concerned with impossibilities and in Section 4.2 we provide protocols. Figure 3 shows a graphical representation of the results.

4.1 Impossibilities

In this section we provide impossibility results for the Byzantine model. Clearly the impossibilities proved for the crash model still hold. In particular the impossibilities for $\mathcal{SC}$(SV1) and $\mathcal{SC}$(WV1) are directly derived from the corresponding ones for the crash model. Next we provide additional impossibilities.

Lemma 4.1 *In the Byzantine model, there is no protocol that solves $\mathcal{SC}(k,t,$WV2$)$, for $t \geq \frac{k}{2k+1}n$ and $t \geq k$.*

Proof: For a contradiction assume that such a protocol A exists. We distinguish two cases: (i) $t \geq n/2$ and (ii) $t < n/2$.

Consider case (i). Let $v_1, v_2, ..., v_{t+1}$ be $t+1$ different values. Let α be the following run of A. The number of actual failures in α is $f = n-t-1$. Let F be the set of faulty processes and let $p_1, ...p_{t+1}$ be the correct processes. Process p_i has input v_i, for $i = 1, 2, ..., t+1$. 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 $i = 1, 2, ..., t+1$ consider the following run α_i. All processes are correct, all have input v_i, 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 $i = 1, 2, ..., t+1$, 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 $v_1, v_2, ..., v_{t+1}$ are different, we have that $t+1$ values are decided in α. But $t \geq k$, hence at least $k+1$ values are decided in α.

Consider case (ii). Since $t < n/2$ we have that $n-2t > 0$ and thus the condition $t \geq \frac{k}{2k+1}n$ is equivalent to $\frac{n-t}{n-2t} \geq k+1$. Then, we can partition the processes into $k+2$ groups, the first $k+1$ of which, denoted $g_1, g_2, ..., g_{k+1}$, each consists of at least $n-2t$ processes, and the last of which, denoted F, consists of t processes. Let α be the following run of A. Let $v_1, v_2, ..., v_{k+1}$ be $k+1$ different values. Processes in g_i start with v_i, for $i = 1, 2, ..., k+1$, 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 $i = 1, 2, ..., k+1$ consider the following run α_i. All processes are correct, all have input v_i, 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 $i = 1, 2, ..., k+1$, 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 $v_1, v_2, ..., v_{k+1}$ are different, we have that $k+1$ values are decided in α. □

Lemma 4.2 *In the Byzantine model, there is no protocol that solves $\mathcal{SC}(k,t,$RV1$)$.*

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 $v_1, ..., v_z$ 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 for some $i \in \{1, ..., z\}$. 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 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

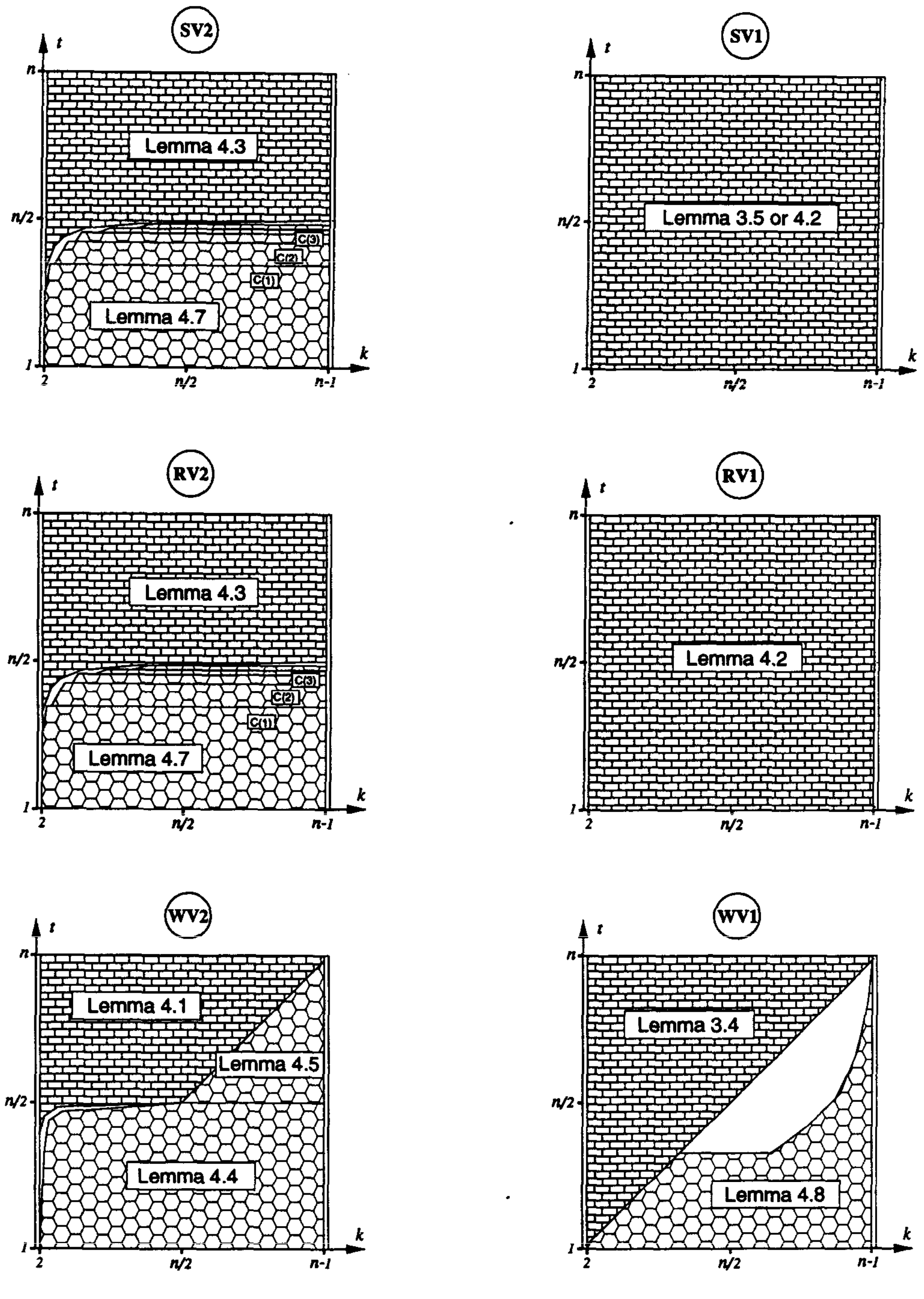

Figure 3: Byzantine model. Regions filled in brick pattern indicate impossibility. Regions filled in honeycomb pattern indicate solvability. Unfilled regions indicate open problems. Figures are drawn to scale $n = 64$.

between α_1 and α_2 they have to decide on the same value. We now distinguish three possible cases: (1) q is different from both p_1 and p_2, (2) q is p_1 and (3) q is 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 (resp. p_2) then p_2 (resp. 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 $\mathcal{SC}(k,t,\text{RV1})$. □

Lemma 4.3 *In the Byzantine model, there is no protocol for $\mathcal{SC}(k,t,\text{RV2})$, for $t \geq \frac{k}{2(k+1)}n$.*

Proof: The proof is similar to that for Lemma 3.6. For a contradiction assume that such a protocol A exists. We distinguish two cases: (i) $t < n/2$ and (ii) $t \geq n/2$. Consider case (i). 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 (ii). In this case we partition the processes in $k+1$ groups each consisting of at least one process.

In both cases, let $g_1, g_2, ..., g_k, g_{k+1}$ be the $k+1$ groups of processes. Let $v_1, ...v_{k+1}$ be $k+1$ different values and consider the following run α. 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 $\mathcal{SC}(k,t,\text{RV2})$, 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. □

4.2 Protocols

In this section we provide protocols for the Byzantine model. We start by observing that PROTOCOL A, used for the crash model, solves $\mathcal{SC}(\text{WV2})$ also in the Byzantine model, though only for a restricted range of values of k and t.

Lemma 4.4 PROTOCOL A *solves $\mathcal{SC}(k,t,\text{WV2})$ in the Byzantine 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 4.5 PROTOCOL A *solves $\mathcal{SC}(k,t,\text{WV2})$ in the Byzantine 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: (i) $f \leq n - t - 2$ and (ii) $f > n - t - 2$. In case (i) 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 (i). In case (ii) the number of correct processes is strictly less than $t+2$. Hence we cannot have more than $t+1$ different decisions. Putting together the two cases, we have 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 [4], 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 $\mathcal{SC}(\text{SV2})$.

> ℓ-`echo protocol:` To ℓ-echo broadcast a message m, the sender s sends the message $\langle$`init`$,s,m\rangle$ to all other processes. When a process p receives the first $\langle$`init`$,s,m\rangle$ from s, it sends the message $\langle$`echo`$,s,m\rangle$ to all other processes. Subsequent `init` messages from s are ignored. If process p receives message $\langle$`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 4.6 *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:*

(i) Correct processes accept at most ℓ different messages.

(ii) If s is correct, every correct process accepts m.

Proof: First we prove (i). By sake of contradiction assume that correct processes accept $\ell+1$ different messages $m_1, m_2, ..., m_{\ell+1}$. Then there must be $\ell+1$ correct processes, say $p_1, p_2, ..., p_{\ell+1}$, such that process p_i receives more than $(n+\ell t)/(\ell+1)$ echos with m_i, for each $i = 1, 2, ..., \ell+1$. Thus there must be a total of more than $n + \ell t$ echos sent for the messages $m_1, m_2, ..., m_{\ell+1}$. 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 and so on) 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 (*ii*). 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. Because there are at most $t \leq (n+\ell t)/(\ell+1)$ faulty processes, 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 $\mathcal{SC}(k,t,\text{SV}2)$ 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 4.7 PROTOCOL C(ℓ) *solves* $\mathcal{SC}(k,t,\text{SV}2)$ *in the Byzantine 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, 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. By the protocol it is necessary that there be k sets $g_1, g_2, ..., g_k$, each consisting of at least $n-2t$ processes, such that some correct process accepts a value v_i from each process in g_i (with $v_i \neq v_j$ for $i \neq j$). Hence correct processes accept at least $k(n-2t)$ values broadcast by ℓ-echo. Each faulty process can contribute ℓ different values, and so 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 $\mathcal{SC}(\text{WV}1)$.

PROTOCOL D: Processes $p_1, p_2, ..., p_{t+1}$ each broadcasts its input value. A process that receives a value v_i from p_i, $i \in \{1,2,...,t+1\}$, broadcasts an $\langle \text{echo}, v_i, p_i \rangle$ message and never echos a value for p_i again. Processes $p_1, p_2, ..., p_k$ each 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 echos $\langle \text{echo}, v_i, p_i \rangle$ 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

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

and

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

Lemma 4.8 PROTOCOL D *solves* $\mathcal{SC}(k,t,\text{WV}1)$ *in the Byzantine model for* $k \geq Z(n,t)$.

Proof: We start by proving termination. At least one process among $p_1, ..., p_{t+1}$ 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: (*i*) $n-t-f \leq 0$ and (*ii*) $n-t-f > 0$. In case (*i*) 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 (*ii*) 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 $p_1, ..., p_{t+1}$ 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 over all 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 Figure 3).

4.3 Remarks

For the Byzantine 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 $\mathcal{SC}(\text{WV}1)$.

5 Conclusions and Future Work

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 [5] and considered by several papers in the literature. Several of 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 had 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 model with either crash-failures or 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.

One area of ongoing work is to perform a similar analysis for k-set consensus problems in shared-memory systems. Many of the techniques we developed here are useful for providing protocols and proving impossibilities in this case. In some cases, translations between the shared memory model and the message passing model are possible [2], allowing us to use the protocols developed for the message passing model. However, these do not always provide solutions with the highest fault tolerance possible, and do not hold when Byzantine failures are possible. In these cases, protocols designed specifically for this context are required. The impossibility result of [3, 7, 13, 1] (i.e., Lemma 3.2) also applies to the shared memory model. Future work will provide full analysis of this context.

A second natural direction is to consider synchronous systems. For synchronous settings, we note that any protocol that works in asynchronous systems works also in synchronous systems. There are protocols to solve $SC(k,t,\text{RV1})$ for any t in the synchronous crash setting (see, e.g., [10, Ch. 7]). Some of the impossibility proofs provided in this paper still work in the synchronous setting. Putting together these results we can almost demarcate the line between possible and impossible for the synchronous message-passing crash model. However for the other models, especially with Byzantine failures, nothing is known (except, of course, for the known results on 1-consensus, e.g., [8, 9, 12]).

References

[1] H. Attiya. A direct proof of the asynchronous lower bound for k-set consensus. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing*, page 314, July 1998.

[2] H. Attiya, A. Bar-Noy and D. Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM* 42(1):124-142, January 1995.

[3] 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, 1993.

[4] G. Bracha and S. Toueg. Resilient consensus protocols. In *Proceedings of the 2nd ACM Symposium on Principles of Distributed Computing*, pages 12-26, 1983.

[5] S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation* 105(1):132-158, July 1993.

[6] M. Fischer, N. Lynch and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM* 32(2):374-382, April 1985.

[7] M. Herlihy and N. Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 111-120, 1993.

[8] L. Lamport. The weak Byzantine generals problem. *Journal of the ACM* 30(3):254-280, 1983.

[9] L. Lamport, R. Shostak and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4(3):382-401, July 1982.

[10] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Francisco, California, 1996.

[11] G. Neiger. Distributed consensus revisited. *Information Processing Letters* 49(4):195-201, 1994.

[12] M. Pease, R. Shostak and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM* 27(2):228-234, April 1980.

[13] M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 101-110, 1993.