Lower Bound on the Number of Rounds for Synchronous Consensus with Dependent Process Failures *

Flavio P. Junqueira
flavio@cs.ucsd.edu

Keith Marzullo
marzullo@cs.ucsd.edu

University of California, San Diego
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA

*This work was developed in the context of the RAMP project, supported by DARPA as project number N66001-01-1-8933.

1 Introduction

In this paper, we present the lower bound on the number of rounds to solve Consensus for a synchronous system in which processes may fail dependently. Processes are said to fail dependently if the failure of one indicates an increase in the failure probability of another process. Failure correlations can be due to intrinsic properties of the system. For example, two processes may fail dependently because they use the same software version containing the same bugs.

To describe these failure correlations, we present two new abstractions: cores and survivor sets. These abstractions are not coupled to any particular failure model and they constitute a generic tool for the design of fault-tolerant algorithms.

The Consensus problem briefly consists in reaching agreement among all correct processes. Every process starts with an initial value, and eventually every correct process decides upon the same value v. The lower bound on the number of rounds for Consensus we show is generic. It applies to any system model in which Consensus is solvable. It just requires the knowledge of the lower bound on process replication, and a system described in terms of cores and survivor sets. To show this result, we use the technique to prove lower bounds described by Keidar and Rajsbaum [1]. This technique is based on the notion of layering.

In the next sections, we introduce our new abstractions and our assumptions for the system. Finally, in the last section we show our main result.

2 System Model

A system is composed of a set $\Pi = \{p_1, p_2, \cdots, p_n\}$ of processes that communicate by exchanging messages. In our model, process failures are allowed to be correlated, meaning that the failure of a process may increase the failure probability of another process.

Assuming that faulty processes do not recover, there has to be at least one correct process in every execution in order to achieve fault-tolerance. We hence distinguish subsets of processes such that the chance of all processes in each of these subsets failing is negligible. Moreover, these subsets are minimal in that removing any process of such a subset c makes the probability of all the processes in c failing non-negligible. We call such subsets *cores*. Cores can be extracted from the information about process failure correlations. For example, suppose a system where processes are represented by attributes. In such

a system, sharing attributes may indicate failure correlation and cores can be computed accordingly. In this paper, however, we assume that the set of cores is provided as part of the system specification. Models to describe failure correlations and methods to extract cores from instances of these models are not addressed here.

By assumption, each core contains at least one process that is correct in some execution. Thus, a subset of processes, such that the intersection with every core is non-empty contains all the processes that are correct in some execution. If such a subset is minimal, then it is called a survivor set. Notice that in every run of the system there is at least one survivor set that contains only correct processes. Our definition of survivor sets is analogous to the one of a *fail-prone system* $\mathcal{B}$ [2]: the set of survivor sets is the complement of $\mathcal{B}$.

We now define cores and survivor sets more formally. Let R be a rational number expressing the target degree of reliability for Π, and $r(x)$, $x \subseteq \Pi$, be a function that evaluates to the reliability of the subset x. We define cores and survivor sets as follows:

Definition 2.1 Given a set of processes Π and target degree of reliability $R \in [0,1] \cap Q$, c is said to be a *core* if and only if:

1. $c \subseteq \Pi$;
2. $r(c) \geq R$;
3. $\forall p \in c,\ r(c - \{p\}) < R$.

Given a set of processes Π and a set of cores C_Π, s is said to be a survivor set if and only if:

1. $s \subseteq \Pi$;
2. $\forall c \in C,\ s \cap c \neq \emptyset$;
3. $\forall p_i \in s,\ \exists c \in C_\Pi$ such that $p_i \in c$ and $(s - \{p_i\}) \cap c = \emptyset$.

We define C_Π and S_Π as the set of cores and the set of survivor sets of Π respectively.

The function $r(.)$ and the target degree of reliability R are used at this point only to formalize the idea of a core. In reality, reliability does not need to be expressed as probabilities. If this information is known by other means, as by the utilization of descriptions based on attributes for instance, then cores can be extracted without using probabilities.

For example, suppose a six process system with $\Pi = \{ph_1, ph_2, pl_1, pl_2, pl_3, pl_4\}$. In this system, ph_1 and ph_2 are very reliable and each of these fail independently of every other $p \in \Pi$. Processes pl_i, for $1 \leq i \leq 4$, however, fail dependently among each other. That is, for every pair of processes pl_i, pl_j, $1 \leq i, j \leq 4$ and $i \neq j$, we have that if pl_i is faulty in some execution of the system, then pl_j is also faulty. Thus, a subset with maximum reliability contains processes ph_1, ph_2, and exactly one process pl_i. Suppose that the maximum reliability achievable for a subset of processes satisfies the intuitive notion of target degree of reliability for this system. We can therefore infer that for each i, $1 \leq i \leq 4$, $\{ph_1, ph_2, pl_i\}$ is a core. The set C_Π of cores is hence as follows:

$$C_\Pi = \{\{ph_1, ph_2, pl_1\}, \{ph_1, ph_2, pl_2\}, \{ph_1, ph_2, pl_3\}, \{ph_1, ph_2, pl_4\}\} \quad (1)$$

From C_Π, it can be easily verified that the set S_Π of survivor sets in this example is as follows:

$$S_\Pi = \{\{ph_1\}, \{ph_2\}, \{pl_1, pl_2, pl_3, pl_4\}\} \quad (2)$$

In the remainder of this paper, we assume that these subsets are provided as part of the system representation. In the following sections, a system is described by a triple $\langle \Pi, C_\Pi, S_\Pi \rangle$, for Π being a set of processes, C_Π being the set of cores of Π, and S_Π being the set of survivor sets of Π. From this point on, we call $\langle \Pi, C_\Pi, S_\Pi \rangle$ a system representation.

3 Process Failures and Replication Requirements

Expressing correlated failures by using cores and survivor sets is not restricted to a particular type of process failure. Here, we assume two different models for process failures: crash model and byzantine model. In the former model, processes fail by crashing. That is, once a process fails, it stops sending and receiving messages. The second model assumes that processes can fail arbitrarily. Examples of arbitrary failures are selective forwarding of messages and arbitrary changes to the content of messages.

In the crash model, we proved in [3] that the following property for process replication is sufficient to solve Consensus:

Property 3.1 $C_\Pi \neq \emptyset$

To solve Consensus in the arbitrary model, a stronger property is necessary for process replication. The following two properties are necessary and sufficient to solve Consensus in such a model, and we prove in [3] that they are equivalent:

Property 3.2 (**Byzantine Partition**) *For every partition* (A, B, C) *of* Π*, at least one of* A*,* B*, or* C *contains a core.*

Property 3.3 (**Byzantine Intersection**) $\forall s_i, s_j \in S_\Pi$*,* $\exists c_k \in C_\Pi$*, such that* $c_k \subseteq (s_i \cap s_j)$*.*

Consider a system represented by $\langle \Pi, C_\Pi, S_\Pi \rangle$. Independently of the model assumed for process failures, there may be more processes than necessary. For example, in the crash model, one core is sufficient. There may be, however, more cores available in the system. In particular, if the goal of the system is to solve another problem and Consensus is being used as a primitive, then this extra amount of replication perhaps is necessary. Thus, we define a subsystem as a subset of processes of the original system that can be used to solve Consensus. In the crash model, a subsystem is a subset of processes in some core, whereas in the arbitrary model, a subsystem is such that it satisfies Byzantine Intersection.

More formally, let Λ be some property that defines the process replication requirement for a given failure model. For example, the byzantine intersection property is the lower bound on process replication in a failure model assuming arbitrary failures and reliable channels. A subsystem is then defined as follows:

Definition 3.4 Let l be a replication requirement and $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ be a system representation. A system sys' represented by $\langle \Pi', C'_\Pi, S'_\Pi \rangle$ is a subsystem of sys if and only if $\Pi' \subseteq \Pi$ and $C'_\Pi \subseteq C_\Pi$ and sys' satisfies Λ.

Under this definition, subsystem is minimal if the removal of a single process or core is such that the lower bound property on process replication is no longer satisfied. More formally, a subsystem $sys' = \langle \Pi', C'_\Pi, S'_\Pi \rangle$ is minimal if and only if there is no other subsystem $sys'' = \langle \Pi'', C''_\Pi, S''_\Pi \rangle$ such that $|\Pi''| < |\Pi'|$ or $|C''_\Pi| < |C'_\Pi|$. This definition is extensively used in Section 5.

4 Synchronous systems

Synchronous systems impose bounds on message delay, process speed, and clock drift. These bounds, however, are not necessarily based on absolute time. As in the model of Dolev *et al.* [4], steps of an algorithm are used to define these bounds. In a step a process may: 1) receive a message; 2) undergo a state transition; 3) send a message to a single process.

Following this model, the timing assumptions for a synchronous system are given by two parameters: $\Phi \geq 1$ and $\Delta \geq 1$. Furthermore, any execution of an algorithm α in such a system satisfies the following properties:

Process synchrony : for any finite subsequence w of consecutive steps, if some process p_i takes $\Phi + 1$ consecutive steps in w, then any process that is still alive at the end of w has taken at least one step in w;

Message synchrony : for any pair of indices k, l, with $l \geq k + \Delta$, if message m is sent to p_i during the k-th step, then m is received by the end of the $l - th$ step.

From these properties, an execution is further organized in rounds, which are defined in terms of steps of processes. In a round, a process p_i executes $n + k$ steps. The first n steps are used by p_i to send real messages, whereas in the last k rounds it sends *null* messages. These k last steps are necessary to guarantee that all messages sent to p_i in a round r are received before p_i proceeds to round $r+1$. The number k of steps is a function of Δ, Φ, n, and r.

5 Lower bound on the Number of Rounds

To demonstrate the lower bound for the synchronous model with dependent failures, we use the technique of layering proposed by Keidar and Rajsbaum [1]. The general idea is to show that the application of environment actions to some initial state still results in states in which alive processes cannot decide. An environment action is exemplified by a process crashing.

A layering is defined as a set of environment actions that can be performed by the system. The set of possible actions is coupled to the failure model assumed. In our case, we assume that a layering consists of crashing at most one process at a round. Before failing in a round, a process is allowed to send messages to a number of process. We assume that every process p_i sends at most one message to another process p_j at each round. We then use $(i, [j])$ to denote that process p_i fails during this round, but the messages p_i sent to processes $\{p_1 \cdots p_j\} \subseteq \Pi$ are received.

A layer are applied to a state. If x is some state, then we denote the application of a layer l to x as $x \cdot l$. We define a state as a string of entries, one per process. Each entry contains the values that compose the local state of a process at a given round. If

some process p_i is crashed at round r, then the state of p_i is represented by a special symbol denoting that it has crashed. For the Consensus problem, every process begins an execution with a initial value. We assume without loss of generality that the set of possible decision values is binary. Thus, for every binary string w of length $|\Pi| = n$, there is a initial state x_w, and $Init$ is the set of all possible initial states. Note that a layer $(i, [j])$ is only applicable to some state x if p_i is not crashed in x.

We call an execution the application of a sequence of layers to some initial state x. More formally, if x is a string representing the initial state of the processes and $l_1 l_2 \cdots l_k$ is a sequence of k distinct layers of $\mathbf{L}$, then $((\cdots((x \cdot l_1) \cdot l_2) \cdots) \cdot l_k)$ is an execution.

Let $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ be a system representation. For this system, let $\Gamma \subseteq \Pi$ be a subset of processes such that there is some execution in which all processes in Γ are faulty and $|\Gamma| = |\Pi| - \min\{|s| : S \in S_\Pi\}$. Observe that $|\Gamma|$ is the maximum number of failures among all valid executions. We hence define the following layering for our model:

$$\mathbf{L} = \{(p, [q]) \mid p \in \Gamma,\ [q] = \{1 \cdots q\} \subseteq \Pi\}$$

We use $\mathbf{L}(x) = \{x \cdot l | l \in L\}$ to denote the application of layering $\mathbf{L}$ to state x and $L(X) = \{L(x) | x \in X\}$ to express the application of layering $\mathbf{L}$ to the set of states X. In addition, we define $\mathbf{L}^i$ as the application of $\mathbf{L}$ for i consecutive times. This is expressed recursively as follows:

$$\begin{aligned} \mathbf{L}^0(X) &= X \\ \mathbf{L}^k(X) &= \mathbf{L}(\mathbf{L}^{k-1}(X)) \end{aligned}$$

We observe, however, that we can have no more than $\kappa = |\Gamma|$ layering applications, where κ is the maximum number of process failures. Thus, the system configuration restricts the number of consecutive applications of $\mathbf{L}$.

Another important definition is the one of similar states. Similarity of states captures the notion of states in which a correct process cannot make a decision, because there is not sufficient information for it to do so. This notion is used extensively in the proofs presented below. Similar states and similarity connected sets of states are defined as follows:

Definition 5.1 States x and y are similar, denoted $x \sim y$, if there is a process p_j that is non-failed in these states, such that (a) x and y are identical except in the local state of p_j, and (b) there exists $p_i \neq p_j$ that is non-failed in both x and y. A set of states is

similarity connected if for every $x, y \in X$ there are states $x = x_0, x_1, \cdots, x_m = y$ so that $x_i \sim x_{i+1}$, for all $0 \leq i \leq m$.

We show now that $Init$ is similarity connected with the following lemma.

Lemma 5.2 *Init is similarity connected.*

Proof:

Given a state z, we denote by z_j the local state of process p_j in the state z. Let y, y' be two states in $Init$. For every $0 \leq m \leq n$, define x^m by setting $x^m_j = y_j$ for all $j > m$ and $x^m_j = y'_j$ for all $j \leq m$. We get: $x^0 = y$ and $x^n = y'$. Note that x^{m-1} and x^m differ exactly in the local state of process p_m. Since all the processes are non-failed in every state in $Init$, these states are similar, that is, $x^{m-1} \sim x^m$.

□

Now, we need to show that any $k \leq \kappa$ consecutive applications of layering $\mathbf{L}$ on a similarity connected set of states generates another similarity connected set of states. With the following lemma, we show that after κ layering applications on a similarity set of states we still have a similarity connected set of states.

Lemma 5.3 *Let X be a similarity connected set of states in which no process is failed and there are at least two correct processes. $L^k(X)$ is similarity connected for all $k \leq \kappa$.*

Proof: We prove by induction. The base case is $k = 0$. By definition, we have that $\mathbf{L}^0(X) = X$. Consequently, $\mathbf{L}^0(X)$ is similarity connected. The induction hypothesis is that $\mathbf{L}^{k-1}(X)$ is similarity connected and we want to show that $\mathbf{L}(\mathbf{L}^{k-1}(X))$ is also similarity connected. To show this, we need to demonstrate that the two following properties hold:

1. if $x \in \mathbf{L}^{k-1}(X)$ then $\mathbf{L}(x)$ is similarity connected;
2. if $y, y' \in \mathbf{L}^{k-1}(X)$, $y \sim y'$, then $\mathbf{L}(y) \cup \mathbf{L}(y')$ is similarity connected;

1: Suppose we apply layers $(i, [0])$ and $(j, [0])$ to x. Because no process is failed in none of these layers, we have that $x \cdot (i, [0])$ and $x \cdot (j, [0])$ are identical. Now let us apply layers $(i, [l-1])$ and $(i, [l])$ to x. $x \cdot (i, [l-1])$ and $x \cdot (i, [l])$ are either identical, in the case that process i did not send a message to l, or differ on the state of l, in which case they are similar.

2: y and y' differ in the state of one process, let's say i. If we apply layer $(i,[n])$ to both states, we get $y \cdot (i,[n])$ and $y' \cdot (i,[n])$. Notice that in this round, no process received a message from i. Moreover, all processes besides i have identical state in y and y' and consequently the messages they send have to be the same. Therefore, we have that $y \cdot (i,[n]) \sim y' \cdot (i,[n])$. Along with property 1, this proves our claim that $\mathbf{L}(y) \cup \mathbf{L}(y')$ is similarity connected.

□

We use the two previous lemmas to show a theorem that provides the lower bound on the number of rounds. The theorem is as follows:

Theorem 5.4 *Let* $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ *be a synchronous system representation,* $sys' = \langle \Pi', C'_\Pi, S'_\Pi \rangle$ *be the representation of a minimal subsystem of sys,* $\mathcal{A}$ *be a Consensus algorithm, and* $\kappa = |\Pi'| - \min\{|s| : S \in S'_\Pi\}$. *If* $|\Pi| - \kappa > 1$, *then there is an execution of* $\mathcal{A}$ *in which* $f \leq \kappa$ *processes are faulty and some correct process takes at least* $f+1$ *rounds to decide.*

Proof: By lemma 5.2, the set of initial states is similarity connected. According to 5.3, the f-th application of layering $\mathbf{L}$ on the set of initial states $Init$ results in another similarity connected set of states. Thus, there is some execution in which after f rounds there is at least one correct process that has not yet decided. We conclude that at least $f+1$ rounds are required for all correct processes to decide.

□

From this theorem, we can extract the following corollary.

Corollary 5.5 *Let* $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ *be the representation of a synchronous system with crash failures,* $sys' = \langle \Pi', C'_\Pi, S'_\Pi \rangle$ *be the representation of a minimum subsystem of sys,* $\mathcal{A}$ *be a Consensus algorithm. If* $|\Pi'| < |\Pi|$, *then there is an execution of* $\mathcal{A}$ *with* $f+1 \leq |c_{\min}|$ *in which some correct process takes at least* $f+1$ *rounds to decide, where* $c_{\min} \in C_\Pi$ *is a minimum-sized core.*

Proof: In the crash model, a core is sufficient to solve Consensus. Thus, a minimal subsystem is composed of a single smallest core. Every survivor set in such a subsystem has size 1. From theorem 5.4, $\kappa = |c_{min}| - 1$ for *sys*. Consequently, for any algorithm, there is an execution in which some correct process takes at least $f+1$ rounds to decide, $f+1 \leq |c_{min}|$.

□

Now we show a theorem that determines the lower bound on the number of rounds in the case that there are executions with a single correct process. To prove this theorem, we use the notation $Crashed_\alpha(r)$ for the set of processes that have failed by round r in execution α.

Theorem 5.6 *Let* $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ *be a synchronous system representation,* $sys' = \langle \Pi', C'_\Pi, S'_\Pi \rangle$ *be the representation of a minimal subsystem of sys,* $\mathcal{A}$ *be a Consensus algorithm, and* $\kappa = |\Pi'| - 1$. *If* $|\Pi| - \kappa = 1$, *then there is an execution of* $\mathcal{A}$ *in which* $f \leq \kappa$ *processes are faulty and some correct process takes at least* $min(\kappa, f+1)$ *rounds to decide.*

Proof:

Suppose that $0 \leq f \leq |\Pi| - 2$. By lemma 5.3, if there are at least two correct processes, then there is at least one execution in which some correct process requires at least $f + 1$ rounds to decide.

Lemma 5.3 does not include the case in which $f = |\Pi| - 1$. We hence show this case separately. We provide a contradiction argument to show that at least $f = \kappa = |\Pi| - 1$ rounds are necessary for a correct process to decide in this case. Suppose that there exists a Consensus algorithm $\mathcal{A}'$ such that in every execution of $\mathcal{A}'$ with $|\Pi| - 1$ faulty processes, no correct process decides later than round $f - 1 = |\Pi| - 2$. Let α be an execution such that $f = |\Pi| - 1$, p_i is the only correct process in α, and p_j is a process that crashes at round $|\Pi| - 1$. More formally, if x_α is the initial state of α, then α is defined as $((\cdots((x_\alpha \cdot l_1) \cdot l_2) \cdots l_{|\Pi|-2}) \cdot l_{|\Pi|-1})$, where: 1) $l_i, l_j \in \mathbf{L}$, l_i and l_j denote distinct faulty processes if $i \neq j$; 2) $l_{|\Pi|-1} = (j, [k])$, $k \in \{1, \cdots, n\}$. From the previous assumption, the single correct process p_i has to decide no later than round $r' \leq |\Pi| - 2$.

Let β be an execution defined as $(\cdots((x_\alpha \cdot l_1) \cdot l_2) \cdots l_{|\Pi|-2})$. Execution β hence is identical to α up to round $|\Pi| - 2$ and p_j is correct in β. Process p_i cannot distinguish execution α from execution β, and consequently p_i and p_j have to decide upon the same value in β. Note that we assumed an arbitrary execution α in which some process p_j fails at round $|\Pi| - 1$. For every such execution, there is an execution β in which p_j is correct and p_i cannot distinguish from α. Because a correct process cannot distinguish α from β, it has to decide at most at round $|\Pi| - 2$. Thus, in every execution of $\mathcal{A}'$ with $f = |\Pi| - 2$, a correct process decides in at most $f = |\Pi| - 2$. By lemmas 5.2 and 5.3, however, $|\Pi| - 2$

applications of layering $\mathbf{L}$ on some initial state x still results in a similarity connected set of states. Thus, such an algorithm $\mathcal{A}'$ cannot exist. We conclude that for every Consensus algorithm $\mathcal{A}$ there is some execution with $f = |\Pi| - 1$ in which a correct process does not decide at round $r < f$.

For every Consensus algorithm $\mathcal{A}$ assuming a system such that $|\Pi| - \kappa = 1$, there is therefore some execution of $\mathcal{A}$ in which some correct process does not decide earlier than $\min(f + 1, |\Pi| - 1)$.

□

6 Final Remarks

For the crash model, a single core is sufficient to solve Consensus, as we observed in Section 3. Consider a system $sys = \langle \Pi, C_\Pi, S_\Pi \rangle$ that contains at least one core. A minimal subsystem sys' of sys is consequently composed of a single minimum-sized core c_{min}. That is, $\Pi' = c_{min}$ and $C_\Pi = \{c_{min}\}$. By the definition of core, at least one process is correct in every execution. The set S'_Π is hence defined by $\{\{p_i\} : p_i \in c_{min}\}$. Because every survivor contains exactly one process, we have that $\kappa = 1$.

Assuming arbitrary process failures, a system configuration has to satisfy Byzantine Intersection so that a solution for Consensus exists. For such a system, the size of a minimum-sized survivor set for a minimal subsystem is not in general one. The single case in which a system has a survivor set with exactly one process and still satisfies Byzantine Intersection is the one of a single reliable process. Such a system is represented by $\langle p_i, \{\{p_i\}\}, \{\{p_i\}\} \rangle$. Every other case has to be such that a minimum-sized survivor set has size at least two. Thus, in general, the lower bound on the number rounds in the worst case differs between the two failure models. We illustrate this concept with the following system representation:

Example 6.1 :

- $\Pi = \{p_a, p_b, p_c, p_d, p_e\}$
- $C_\Pi = \{\{p_a, p_b, p_c\}, \{p_a, p_d\}, \{p_a, p_e\}, \{p_b, p_d\}, \{p_b, p_e\}, \{p_c, p_d\}, \{p_c, p_e\}, \{p_d, p_e\}\}$
- $S_\Pi = \{\{p_a, p_b, p_c, p_d\}, \{p_a, p_b, p_c, p_e\}, \{p_a, p_d, p_e\}, \{p_b, p_d, p_e\}, \{p_c, p_d, p_e\}\}$

For the crash model, a minimal subsystem $\langle \Pi', C'_\Pi, S'_\Pi \rangle$ is such that $|\Pi|' = 2$, $|C'_\Pi| = 1$, and a minimum-sized survivor set contains a single process. By Theorem 5.4, the lower bound

on the number of rounds is 2 in the worst case ($\kappa = 1$ and $|\Pi| - \kappa > 1$). In the arbitrary model, $\langle \Pi, C_\Pi, S_\Pi \rangle$ is already a minimal subsystem: if any process or core is removed, then the remaining subsystem does not satisfy Byzantine Partition. By Theorem 5.4, the lower bound on the number of rounds is 3 in the worst case ($\kappa = 2$ and $|\Pi| - \kappa > 1$).

This is in contrast with the traditional result for Consensus under the the assumption of independent and identically distributed process failures, where the lower bound on the number of rounds in the worst case is the same in both models.

References

[1] I. Keidar and S. Rajsbaum, "On the Cost of Fault-Tolerant Consensus When There Are No Faults - A Tutorial," Tech. Rep. MIT-LCS-TR-821, MIT, May 2001.

[2] D. Malkhi and M. Reiter, "Byzantine Quorum Systems," in *29th ACM Symposium on Theory of Computing*, pp. 569–578, may 1997.

[3] F. Junqueira and K. Marzullo, "Consensus for Dependent Process Failures," tech. rep., UCSD, La Jolla, CA, September 2002. http://www.cs.ucsd.edu/users/flavio/Docs/Gen.ps.

[4] D. Dolev, C. Dwork, and L. Stockmeyer, "On the Minimal Synchronism Needed for Distributed Consensus," *Journal of the ACM*, vol. 1, pp. 77–97, January 1987.