Tight Bounds for Asynchronous Randomized Consensus*

(Preliminary Version)

Hagit Attiya
Department of Computer Science
Technion
Haifa, Israel
hagit@cs.technion.ac.il

Keren Censor
Department of Computer Science
Technion
Haifa, Israel
ckeren@cs.technion.ac.il

ABSTRACT

A distributed consensus algorithm allows n processes to reach a common decision value starting from individual inputs. *Wait-free* consensus, in which a process always terminates within a finite number of its own steps, is impossible in an asynchronous shared-memory system. However, consensus becomes solvable using randomization when a process only has to terminate with probability 1. Randomized consensus algorithms are typically evaluated by their *total step complexity*, which is the expected total number of steps taken by all processes.

This work proves that the total step complexity of randomized consensus is $\Theta(n^2)$ in an asynchronous shared memory system using multi-writer multi-reader registers. The bound is achieved by improving both the lower and the upper bounds for this problem.

In addition to improving upon the best previously known result by a factor of $\log^2 n$, the lower bound features a greatly streamlined proof. Both goals are achieved through restricting attention to a set of *layered* executions and using an isoperimetric inequality for analyzing their behavior.

The matching algorithm decreases the expected total step complexity by a $\log n$ factor, by leveraging the multi-writing capability of the shared registers. Its correctness proof is facilitated by viewing each execution of the algorithm as a stochastic process and applying Kolmogorov's inequality.

Categories and Subject Descriptors

D.1.3 [**Software**]: Programming Techniques—*Concurrent programming*; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Stochastic processes*

*This research is supported by the *Israel Science Foundation* (grant number 953/06).

STOC'07, June 11–13, 2007, San Diego, California, USA.

General Terms

Algorithms, Theory

Keywords

distributed computing, shared-memory, lower bound, randomized algorithms, isoperimetric inequality

1. INTRODUCTION

Coordinating the actions of processes is crucial for virtually all distributed applications, especially in asynchronous systems. At the core of many coordination problems is the need to reach *consensus* among processes, despite the possibility of process failures. A (binary) consensus algorithm allows processes starting with input values in $\{0, 1\}$ to agree on the same output value (*agreement*); to rule out trivial solutions, this common output must be one of the inputs (*validity*). Consensus is a fundamental task in asynchronous systems, and can be employed to implement arbitrary concurrent objects [21]; consensus is also a key component of the state-machine approach for replicating services [22, 28].

Perhaps the most celebrated result in distributed computing shows that no deterministic algorithm can achieve consensus in an asynchronous system, if one process may fail [20, 21, 23]. Due to the importance of the consensus problem, much research was invested in trying to circumvent this impossibility result. One successful approach is to allow randomized algorithms in which a nonfaulty process terminates only with probability 1. (The agreement and validity properties remain the same.) Randomized consensus algorithms are typically evaluated by their *total step complexity*, which is the expected total number of steps taken by all processes.

Many randomized consensus algorithms have been suggested, in different communication models and under various assumptions about the adversary (see [4]). In particular, algorithms were designed to solve randomized consensus in asynchronous shared-memory systems, against a *strong* adversary that can observe the results of local coin flips before scheduling the processes [1, 2, 5, 26]. The total step complexity of the best previously known algorithm is $O(n^2 \log n)$ [13]. A lower bound of $\Omega(\frac{n^2}{\log^2 n})$ on the expected total number of coin flips was proved by Aspnes [3]; this implies the same lower bound on the total step complexity.

Closing the gap of $\Theta(\log^3 n)$ between the lower and the upper bounds and determining the total step complexity of randomized consensus remained an intriguing open question.

In this paper, we prove that $\Theta(n^2)$ is a tight bound on the total step complexity of solving randomized consensus under the strong adversary in asynchronous shared memory systems, where processes communicate by reading and writing to multi-writer multi-reader registers.

The $\Omega(n^2)$ lower bound is obtained by considering a restricted set of schedules with a round-based structure, called *layers* [25]. We focus on configurations at the end of each layer and classify them according to their *valence* [20, 25], namely, the decisions that can be reached in layered extensions. As opposed to deterministic algorithms, where the valence of a configuration binds the extension to reach a certain decision value v, in a randomized algorithm the valence only implies that some execution will decide v with high probability [3]. This introduces a category of *null-valent* configurations, from which no decision value is reached with high probability. When a decision is reached, the configuration must be *univalent*, so the proof aims to avoid univalent configurations. An important case in the proof is when the configuration is null-valent, where we derive an isoperimetric inequality in order to control a one-round coin-flipping game for reaching another null-valent configuration.

To show that the lower bound is tight, we present a randomized algorithm for consensus, which has an $O(n^2)$ total step complexity. We give a *shared coin* algorithm with a constant agreement parameter, which leverages a single binary multi-writer register (in addition to n single-writer multi-reader registers); this can be used to obtain a randomized consensus algorithm with the same total step complexity [5]. We prove that all processes output the same value with high probability by viewing any schedule of the algorithm as a stochastic process, and applying Kolmogorov's inequality.

Related Work. The previous $\Omega(\frac{n^2}{\log^2 n})$ lower bound [3] relies on a lower bound for coin flipping games. In this proof, the adversary schedules processes step-by-step, and the results of the games are analyzed through hyperbolic functions, resulting in a long and unwieldy proof. In contrast, our approach considers only the configurations at the end of layers, allowing powerful results about product probability spaces to be applied, and streamlining the analysis of the behavior of executions.

Our general proof structure follows a proof by Bar-Joseph and Ben-Or [12] of an $\Omega(\sqrt{n/\log n})$ lower bound on the expected number of rounds in a randomized consensus algorithm for the *synchronous message passing* model. In particular, like them, we treat null-valent configurations by considering one-round coin-flipping games and applying an isoperimetric inequality. Unlike their proof, our proof handles the more complicated shared-memory model and exploits the fact that in an asynchronous system, processes can be hidden in a one-round coin flipping game without having to fail for the rest of the execution.

The layered approach was introduced by Moses and Rajsbaum [25], who used it to study deterministic consensus. They showed that the layered approach can unify the impossibility proof for asynchronous consensus [20,21,23] with the lower bound on the number of rounds needed for solving synchronous consensus [16, 19]. Their work considered the message-passing model as well as the shared-memory model with *single-writer* registers. We take the layered approach one step further and extend it to randomized algorithms, deriving the lower bound on their total step complexity within the same framework as the results for deterministic algorithms. Besides incorporating randomization into the layered model, our proof also deals with the technical challenge of allowing processes to access *multi-writer* registers.

Abrahamson [1] presented a randomized algorithm for solving consensus in asynchronous systems using shared memory, which had an exponential running time. The first polynomial algorithm for solving randomized consensus was presented by Aspnes and Herlihy [5]. They described an algorithm that uses a shared coin in order to reach agreement and has a total step complexity of $O(n^4)$. The amount of shared memory required by this algorithm was later bounded by Attiya, Dolev and Shavit [7]. Aspnes [2] presented an algorithm for randomized consensus with $O(n^4)$ total step complexity, which also uses bounded space. These algorithms were followed by an algorithm of Saks, Shavit and Woll [26] with $O(n^3)$ total step complexity and later, an algorithm of Bracha and Rachman [13] with $O(n^2 \log n)$ total step complexity.

Organization. In Section 2, we adapt the layered model to incorporate randomization and a strong adversary. Section 3 defines the key notions used in our lower bound proof—*potence* and *valence*—and proves that layered executions in the multi-writer shared-memory model are *potence connected*. The lower bound proof appears in Section 4, while Section 5 presents the algorithm.

2. LAYERED MODEL FOR RANDOMIZED DISTRIBUTED ALGORITHMS

We consider a standard model of an asynchronous shared-memory system, where n processes, $p_1, \ldots, p_n$, communicate by reading and writing to shared multi-writer multi-reader registers (cf. [8, 24]).

Each *step* consists of some local computation, including an arbitrary number of local coin flips (possibly biased) and one shared memory *event*, which is a read or a write to some register.

A configuration C consists of the local states of all the processes, and the values of all the registers. Like many impossibility results, our proof relies on having configurations that are indistinguishable to all processes, except some set P. We denote $C \overset{\neg P}{\sim} C'$ if the state of all processes that are not in P is equal in both C and C', and the values of all the registers are equal. We write $C \overset{\neg p}{\sim} C'$ when $P = \{p\}$.

For the purpose of the lower bound, we restrict our attention to a constrained set of executions, which proceed in layers. An *f-layer* is a sequence of at least $n - f$ distinct process id's. When executing a layer L, each process $p \in L$ takes a step, in the order specified by the layer.

An *f-execution* $\tau = L_1, L_2, \cdots$ is a (finite or infinite) sequence of f-layers. We will consider only configurations that are reachable by finite f-executions, namely, after some finite sequence of f-layers.

Since we consider randomized algorithms, for each configuration C there is a fixed probability for every step a process will perform when next scheduled. Denote by X_i^C the probability space of the steps that process p_i will preform, if scheduled by the adversary. The probability space X_i^C depends only on the local state of p_i in configuration C, and therefore, delaying p_i does not change this probability space. Let $X^C = X_1^C \times X_2^C \times \cdots \times X_n^C$ be the product

probability space. An element $\vec{y} \in X^C$ represents a possible result of the local coin flips from a configuration C.

We assume a *strong* adversary that observes the processes' local coin flips, and chooses the next f-layer knowing what is the next step each process will take. The adversary applies a function σ to choose the next f-layer to execute for each configuration C and $\vec{y} \in X^C$, i.e.,

$$\begin{aligned} \sigma \quad &: \quad \{(C, \vec{y}) \mid C \text{ is a configuration and } \vec{y} \in X^C\} \\ &\rightarrow \quad \{L \mid L \text{ is a layer}\}. \end{aligned}$$

When the configuration C is clear from the context we will use the abbreviation $\sigma(\vec{y}) = L_{\vec{y}}$.

Denote by $(C, \vec{y}, L_{\vec{y}})$ the configuration that is reached by applying steps of the processes in $L_{\vec{y}}$, after a specific $\vec{y} \in X^C$ is chosen. Then $C \circ \sigma$ is a random variable whose values are the configurations $(C, \vec{y}, L_{\vec{y}})$, when $\vec{y}$ is drawn from the probability space X^C.

An *f-adversary* $\alpha = \sigma_1, \sigma_2, \cdots$ is a (finite or infinite) sequence of functions.

Given a configuration C and a finite prefix $\alpha_\ell = \sigma_1, \sigma_2, \cdots, \sigma_\ell$ of the adversary α, $C \circ \alpha_\ell$ is a random variable whose values are the configurations that can be reached by the algorithm. For every $\vec{y}_1 \in X^C$ let $P(\vec{y}_1) = \Pr[\vec{y}_1 \text{ is chosen}]$ denote the probability of $\vec{y}_1$ in the probability space X^C. The probability that a configuration C' is reached is defined inductively[1]:

$$\Pr[C \circ \alpha_\ell \text{ is } C'] = \sum_{\vec{y}_1 \in X^C} P(\vec{y}_1) \cdot \Pr[(C, \vec{y}_1, L_{\vec{y}_1}) \circ \alpha'_\ell \text{ is } C'],$$

where α'_ℓ is the remainder of the prefix after σ_1, i.e., $\alpha'_\ell = \sigma_2, \cdots \sigma_\ell$, and the basis of the induction for $\alpha_1 = \sigma_1$ is:

$$\Pr[C \circ \alpha_1 \text{ is } C'] = \sum_{\vec{y}_1 \in X^C} P(\vec{y}_1) \cdot \ \chi_{C'}(\vec{y}_1),$$

where $\chi_{C'}(\vec{y}_1) = \ \chi_{C'}(C, \alpha_1, \vec{y}_1)$ characterizes whether the configuration C' is reached if $\vec{y}_1$ is chosen, i.e.,

$$\chi_{C'}(\vec{y}_1) = \begin{cases} 1 & (C, \vec{y}_1, L_{\vec{y}_1}) \text{ is } C' \\ 0 & \text{otherwise} \end{cases}$$

The probability of deciding v when executing the algorithm under α from the configuration C is defined as follows: if C is a configuration in which there is a decision v, then $\Pr[\text{decision from } C \text{ under } \alpha \text{ is } v] = 1$, if C is a configuration in which there is a decision $\bar{v}$, then $\Pr[\text{decision from } C \text{ under } \alpha \text{ is } v] = 0$, otherwise,

$$\Pr[\text{decision from } C \text{ under } \alpha \text{ is } v] = \sum_{\vec{y}_1 \in X^C} P(\vec{y}_1) \cdot \Pr[\text{decision from } (C, \vec{y}_1, L_{\vec{y}_1}) \text{ under } \alpha' \text{ is } v],$$

where α' is the remainder of the adversary after σ_1, i.e., $\alpha' = \sigma_2, \sigma_3, \cdots$.

3. POTENCE AND VALENCE OF CONFIGURATIONS

In order to derive our lower bound, we are interested in the probability of reaching each of the possible decision values, from a given configuration. As the algorithm proceeds towards a decision, we expect the probability of reaching a decision to grow, where for a configuration in which a decision is reached, this probability is 1. Let $k \geq 0$ be an integer, and define

$$\epsilon_k = \frac{1}{n\sqrt{n}} - \frac{k}{(n-f)^3}.$$

Our proof makes use of adversaries that have a probability of $1 - \epsilon_k$ for reaching a certain decision value. As the layer number k increases, the value of ϵ_k decreases, and the probability $1 - \epsilon_k$ required for a decision grows.

An adversary with a high probability of deciding is defined as follows.

DEFINITION 1. *An f-adversary α from a configuration C that is reachable from an initial configuration by an f-execution with $k \geq 0$ layers, is* v-deciding *if $Pr[$decision from C under α is $v] > 1 - \epsilon_k$.*

Next, we classify configurations according to the probabilities of reaching each of the possible decisions from them. We adapt the notion of *potence* [25] to fit randomized algorithms.

Instead of considering all possible adversaries, we further restrict our attention to a certain subset of them, which will be specified later.

DEFINITION 2. *A configuration C is* (v, k, S)-potent*, for $v \in \{0, 1\}$ and a set S of f-adversaries, if there is a v-deciding adversary $\alpha \in S$ from C.*

DEFINITION 3. *A configuration is* (v, k, S)-valent *if it is (v, k, S)-potent but not $(\bar{v}, k, S)$-potent. Such a configuration is* (k, S)-univalent.

A configuration is (k, S)-bivalent *if it is both $(0, k, S)$-potent and $(1, k, S)$-potent.*

A configuration is (k, S)-null-valent *if it is neither $(0, k, S)$-potent nor $(1, k, S)$-potent.*

We often say that C is v-potent (v-valent, bivalent, null-valent) *with respect to* S. Figure 1 illustrates the valence of configurations.

Note that a configuration can have certain valency with respect to one set of adversaries S and another valency with respect to another set S'. For example, it can be univalent with respect to S and null-valent with respect to S'; however, this cannot happen when $S \subseteq S'$. (Another example appears in Lemma 1 below.) We will sometimes use the notation v-potent or v-valent, when the set S and the layer number k are clear from the context.

The set of f-adversaries we consider, denoted S_P, is induced by a subset of processes $P \subseteq \{p_1, \ldots, p_n\}$. A layer is *$P$-free*, for some set of processes P, if it does not include any process $p \in P$. An adversary α is in S_P, if all of the layers it may choose are P-free. A layer is *full with respect to* S_P if it contains the $n - |P|$ distinct process identifiers $\{p_1, \ldots, p_n\} \setminus P$; otherwise, the layer is *partial*.

Restricting a set of adversaries can only eliminate possible adversaries, and therefore cannot introduce potence that does not exist in the original set of adversaries, as formalized in the following simple lemma.

LEMMA 1. *If a configuration C is v-valent with respect to S_P, then it is not $\bar{v}$-potent with respect to $S_{P \cup \{p\}}$ for any process p.*

[1]For simplicity, we assume that all the probability spaces are discrete, but a similar treatment holds for arbitrary probability spaces.

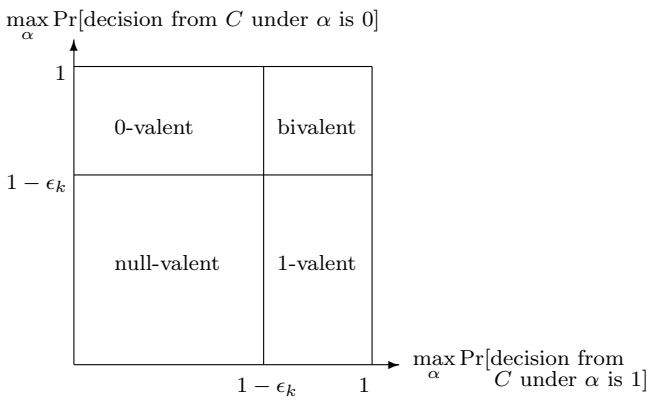

Figure 1: Classifying configurations according to their valence.

PROOF. Assume towards a contradiction, that there is a process p such that C is $\bar{v}$-potent with respect to $S_{P\cup\{p\}}$. Then there exists a $\bar{v}$-deciding adversary α in $S_{P\cup\{p\}}$, i.e.,

$$\Pr[\text{decision in } C' \circ \alpha \text{ is } \bar{v}] > 1 - \epsilon_k.$$

But α is also an adversary in S_P because $S_{P\cup\{p\}} \subseteq S_P$, which implies that C is $\bar{v}$-potent also with respect to S_P, contradicting the fact that C is v-valent with respect to S_P. □

Let C be a configuration, and fix $\vec{y}_1 \in X^C$. We consider the configurations that can be reached by applying different layers. We define a relation between these configurations, based on their potence, which generalizes notions suggested by Moses and Rajsbaum [25].

DEFINITION 4. *For a given $\vec{y}_1$, configurations $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have* shared potence with respect to S_P, *if they are both v-potent with respect to S_P for some $v \in \{0, 1\}$.*

DEFINITION 5. *For a given $\vec{y}_1$, configurations $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ are* potence connected with respect to S_P, *if there is a sequence of layers $L = L_0, L_1, \ldots, L_h = L'$ such that for every i, $0 \leq i < h$, there exists a process p such that the configurations $(C, \vec{y}_1, L_i)$ and $(C, \vec{y}_1, L_{i+1})$ have shared potence with respect to $S_{P\cup\{p\}}$.*

Note that potence connectivity is a transitive relation. Also, if $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have shared potence with respect to $S_{P\cup\{p\}}$ for some process p, then, in particular, they are potence connected.

The following claims show specific configurations that are potence connected, under the assumption that for every process p and layer L, the configuration $(C, \vec{y}_1, L)$ is univalent with respect to S_P and $S_{P\cup\{p\}}$. This implies that if $(C, \vec{y}_1, L) \overset{\neg P\cup\{p\}}{\sim} (C, \vec{y}_1, L')$, then $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have shared potence with respect to $S_{P\cup\{p\}}$, since they must have the same valence with respect to $S_{P\cup\{p\}}$, but are not null-valent.

CLAIM 2. *If $L = [p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}]$ is a layer where for some j, $1 \leq j < \ell$, p_{i_j} and $p_{i_{j+1}}$ both write to the same register R, and $L' = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, \ldots, p_{i_\ell}]$ is the layer L after removing p_{i_j}, then $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have shared potence with respect to $S_{P\cup\{p_{i_j}\}}$.*

PROOF. It is clear that $(C, \vec{y}_1, L) \overset{\neg P\cup\{p_{i_j}\}}{\sim} (C, \vec{y}_1, L')$, which implies that $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have shared potence with respect to $S_{P\cup\{p_{i_j}\}}$. □

CLAIM 3. *If $L = [p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}]$ is a layer, p is a process not in L, and $L' = [p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}, p]$ is the layer L after adding p at the end, then $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ have shared potence with respect to $S_{P\cup\{p\}}$.*

PROOF. If p performs a read operation, then $(C, \vec{y}_1, L) \overset{\neg P\cup\{p\}}{\sim} (C, \vec{y}_1, L')$, which implies that these two configurations have shared potence with respect to $S_{P\cup\{p\}}$, and the claim follows.

If p performs a write operation to register R, then the states of all processes not in $P\cup\{p\}$ are the same in $(C, \vec{y}_1, L)$ and in $(C, \vec{y}_1, L')$, but the value of R may be different.

If $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ do not have shared potence with respect to $S_{P\cup\{p\}}$, then since we assume they are univalent with respect to $S_{P\cup\{p\}}$, we have that for some $v \in \{0, 1\}$, $(C, \vec{y}_1, L)$ is v-valent with respect to $S_{P\cup\{p\}}$ and $(C, \vec{y}_1, L')$ is $\bar{v}$-valent with respect to $S_{P\cup\{p\}}$.

In particular, there is a $\bar{v}$-deciding adversary $\alpha \in S_{P\cup\{p\}}$ from $(C, \vec{y}_1, L')$. Adding p at the beginning of α yields a $\bar{v}$-deciding adversary from $(C, \vec{y}_1, L)$, which is in S_P. However, by Lemma 1, $(C, \vec{y}_1, L)$ is v-valent with respect to S_P, which is a contradiction. □

CLAIM 4. *If $L = [p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}]$ is a layer and $L' = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, p_{i_j}, p_{i_{j+2}}, \ldots, p_{i_\ell}]$ is the layer L after swapping p_{i_j} and $p_{i_{j+1}}$, then $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ are potence connected with respect to S_P.*

PROOF. If p_{i_j} and $p_{i_{j+1}}$ access different registers or if they both read, then $(C, \vec{y}_1, L) \overset{\neg P}{\sim} (C, \vec{y}_1, L')$. If p_{i_j} reads register R and $p_{i_{j+1}}$ writes to R, then $(C, \vec{y}_1, L) \overset{\neg P\cup\{p_{i_j}\}}{\sim} (C, \vec{y}_1, L')$. Both cases imply that $(C, \vec{y}_1, L)$ and $(C, \vec{y}_1, L')$ are potence connected with respect to S_P. The remaining case is when p_{i_j} and $p_{i_{j+1}}$ both write to the same register R, which is proved by reverse induction on j.

Basis: If $j = \ell - 1$, let $L_0 = L$, $L_1 = [p_{i_1}, \ldots, p_{i_{\ell-2}}, p_{i_\ell}]$ be the layer L after removing $p_{i_{\ell-1}}$, and $L_2 = L'$. By Claim 2, $(C, \vec{y}_1, L_0)$ and $(C, \vec{y}_1, L_1)$ are potence connected with respect to S_P, and by Claim 3, $(C, \vec{y}_1, L_1)$ and $(C, \vec{y}_1, L_2)$ are potence connected with respect to S_P. By the transitivity of potence connectivity, this implies that $(C, \vec{y}_1, L_0)$ and $(C, \vec{y}_1, L_2)$ are potence connected with respect to S_P.

Induction step: Let

$$L_0 = L = [p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}]$$

and

$$L_1 = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, p_{i_{j+2}}, \ldots, p_{i_\ell}]$$

be the layer L_0 after removing p_{i_j}. By Claim 2, $(C, \vec{y}_1, L_0)$ and $(C, \vec{y}_1, L_1)$ are potence connected with respect to S_P. Let

$$L_2 = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, p_{i_{j+2}}, \ldots, p_{i_\ell}, p_{i_j}]$$

be the layer L_1 after adding p_{i_j} at the end. By Claim 3, $(C, \vec{y}_1, L_1)$ and $(C, \vec{y}_1, L_2)$ are potence connected with respect to S_P.

For every m, $3 \leq m \leq \ell - j + 1$, let

$$L_m = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, p_{i_{j+2}}, \ldots, p_{i_j}, p_{i_{\ell-m+3}}, \ldots, p_{i_\ell}]$$

be the previous layer L_{m-1} after swapping p_{i_j} with the process before it, until it reaches $p_{i_{j+1}}$. Specifically,

$$L_{\ell-j+1} = L' = [p_{i_1}, \ldots, p_{i_{j-1}}, p_{i_{j+1}}, p_{i_j}, p_{i_{j+2}}, \ldots, p_{i_\ell}].$$

By the induction hypothesis, $(C, \vec{y}_1, L_m)$ and $(C, \vec{y}_1, L_{m+1})$ are potence connected with respect to S_P, for every m, $2 \leq m < \ell - j + 1$. This implies that $(C, \vec{y}_1, L_0)$ and $(C, \vec{y}_1, L_{\ell-j+1})$ are potence connected with respect to S_P. □

The following lemma shows that given a configuration C and a $\vec{y} \in X^C$, if there is a layer that extends C into a v-valent configuration and a layer that extends C into a $\bar{v}$-valent configuration, then there is a layer that extends C into a non-univalent configuration, possibly by failing one additional process.

LEMMA 5. *Let C be a configuration and $\vec{y}_1 \in X^C$. If there are layers L_v and $L_{\bar{v}}$, such that $(C, \vec{y}_1, L_v)$ is $(v, k+1, S_P)$-valent and $(C, \vec{y}_1, L_{\bar{v}})$ is $(\bar{v}, k+1, S_P)$-valent, then there is a layer L such that $(C, \vec{y}_1, L)$ is either not $(k+1, S_P)$-univalent or not $(k+1, S_{P\cup\{p\}})$-univalent, for some process p.*

PROOF. Assume towards a contradiction that for every layer L and every process p, the configuration $(C, \vec{y}_1, L)$ is univalent with respect to both S_P and $S_{P\cup\{p\}}$. Let L^F be the full layer with respect to S_P consisting of all processes not in P, according to the order of their id's. Then, L^F is univalent with respect to S_P, say it is $(\bar{v}, k+1, S_P)$-valent. (Otherwise, we follow the same proof with $L_{\bar{v}}$.)

Assume $L_v = [p_{i_1}, \ldots, p_{i_\ell}]$ and consider the layer $L' = [p_{i_1}, \ldots, p_{i_\ell}, \ldots]$ that is full with respect to S_P, and has L_v as a prefix.

We start with the layer L^F and repeatedly swap processes until we reach the layer L', in a chain of configurations which, by Claim 4, are potence connected with respect to S_P. From L', we repeatedly remove the last process until reaching the layer L_v, in a chain of configurations which, by Claim 3, are potence connected with respect to S_P. This implies that $(C, \vec{y}_1, L^F)$ and $(C, \vec{y}_1, L_v)$ are potence connected with respect to S_P.

Since $(C, \vec{y}_1, L_v)$ is $(v, k+1, S_P)$-valent, and $(C, \vec{y}_1, L^F)$ is $(\bar{v}, k+1, S_P)$-valent, it follows that there are layers L_1 and L_2 such that $(C, \vec{y}_1, L_1)$ is $(v, k+1, S_P)$-valent, $(C, \vec{y}_1, L_2)$ is $(\bar{v}, k+1, S_P)$-valent, and $(C, \vec{y}_1, L_1)$ and $(C, \vec{y}_1, L_2)$ have shared potence with respect to $S_{P\cup\{p\}}$ for some process p.

By Lemma 1 and our assumption, $(C, \vec{y}_1, L_1)$ is $(v, k+1, S_{P\cup\{p\}})$-valent, and $(C, \vec{y}_1, L_2)$ is $(\bar{v}, k+1, S_{P\cup\{p\}})$-valent, and hence, they cannot have shared potence with respect to $S_{P\cup\{p\}}$. This yields a contradiction and proves the lemma. □

4. THE LOWER BOUND

A deciding configuration has to be univalent, so our proof aims to avoid univalent configurations. We first show that some initial configuration is not univalent (Lemma 6); namely, it is either bivalent or null-valent.

Ideally, we would like to prove that a non-univalent configuration can be extended by a single layer to a non-univalent configuration, by (permanently) failing at most one more process. Doing so would allow us to construct a layered execution with f layers, each containing at least $n-f$ process steps, which implies the desired lower bound.

However, while this can be done (with high probability) in the case of a null-valent configuration (see Lemma 11), this is not true in the case of a bivalent configuration. From a bivalent configuration, we have both a v-deciding adversary and a $\bar{v}$-deciding adversary. However, we cannot use them in Lemma 5 to obtain a non-univalent configuration, since the first layer of a v-deciding adversary may lead to a $\bar{v}$-valent configuration. Such a $\bar{v}$-valent configuration, which is reached while following a v-deciding adversary, will be called *$\bar{v}$-switching*.

We prove that a bivalent configuration can be extended by a single layer to a non-univalent or switching configuration, by failing at most one more process (Lemma 7). We also prove that there is a small probability of deciding in a switching configuration and thus, the execution can be extended (with high probability) from a switching configuration by at least one layer (Lemma 8).

We extend the execution in this manner, with high probability, for f layers. Since $n-f$ processes take a step in each layer, we obtain the bound of an expected $\Omega(f(n-f))$ steps (Theorem 12).

4.1 Initial Configurations

We start by applying Lemma 1 to prove that some initial configuration is not univalent.

LEMMA 6. *There exists an initial configuration C that is not univalent with respect to either $S_\emptyset$ or $S_{\{p\}}$, for some process p.*

PROOF. Assume that all initial configurations are univalent with respect to $S_\emptyset$. Consider the initial configurations $C_0, C_1, \cdots, C_n$ such that in C_i, $0 \leq i \leq n$, the input of process p_j is 1 if $j \leq i$ and 0, otherwise. By the validity condition, C_0 is $(0, 0, \emptyset)$-valent and C_n is $(1, 0, \emptyset)$-valent. Therefore, there is an i, $1 \leq i \leq n$, such that C_{i-1} is $(0, 0, \emptyset)$-valent and C_i is $(1, 0, \emptyset)$-valent. Clearly, $C_{i-1} \overset{\neg p_i}{\sim} C_i$, and hence, C_{i-1} and C_i have the same valence with respect to $S_{\{p_i\}}$. By Lemma 1, C_{i-1} is not 1-potent with respect to $S_{\{p_i\}}$, and C_i is not 0-potent with respect to $S_{\{p_i\}}$. Hence, they are null-valent with respect to $S_{\{p_i\}}$. □

4.2 Bivalent and Switching Configurations

We formally define switching configurations as follows.

DEFINITION 6. *Let C be a (v, k, S_P)-potent configuration, $\alpha = \sigma_1, \sigma_2, \ldots$ be a v-deciding adversary from C in S_P, and $\vec{y}_1 \in X^C$ such that the configuration $(C, \vec{y}_1, \sigma_1(\vec{y}_1))$ is $(\bar{v}, k+1, S_P)$-valent. Then $(C, \vec{y}_1, \sigma_1(\vec{y}_1))$ is a* $\bar{v}$-switching configuration with respect to S_P from C by $\vec{y}_1$ and α.

Lemma 5 implies that a bivalent configuration can be extended with one layer to a configuration which is either v-switching or not univalent.

LEMMA 7. *If a configuration C reachable by an f-execution is (k, S_P)-bivalent, then there is an adversary σ such that for every $\vec{y}_1 \in X^C$, $(C, \vec{y}_1, \sigma(\vec{y}_1))$ is either v-switching, or not $(k+1, S_P)$-univalent or not $(k+1, S_{P\cup\{p\}})$-univalent, for some process p.*

PROOF. Assume that for every layer L and every process p, the configuration $(C, \vec{y}_1, L)$ is univalent with respect to S_P and $S_{P\cup\{p\}}$.

Consider the extension of C with L^F. Fix $\vec{y}_1 \in X^C$ and assume that $D = (C, \vec{y}_1, L^F)$ is $(\bar{v}, k+1, S_P)$-valent. Since C is bivalent, there is a v-deciding adversary $\alpha = \sigma_1, \sigma_2, \cdots$ in S_P. Consider the configuration $C' = (C, \vec{y}_1, \sigma_1(\vec{y}_1))$. By the assumption, C' is univalent. If it is $(\bar{v}, k+1, S_P)$-valent then it is $\bar{v}$-switching with respect to S_P from C by $\vec{y}_1$ and α. Otherwise, it is $(v, k+1, S_P)$-valent. Since D is $(\bar{v}, k+1, S_P)$-valent, by Lemma 5, there exists a layer L and a process p such that $(C, \vec{y}_1, L)$ is either not $(k+1, S_P)$-univalent or not $(k+1, S_{P\cup\{p\}})$-univalent. □

Handling switching configurations is more delicate, and it is done in the next lemma.

LEMMA 8. *Let C' be a $\bar{v}$-switching configuration with respect to S_P from C by $\vec{y}_1$ and α. Then with probability at least $1 - \frac{1}{n\sqrt{n}}$, C' can be extended with at least one layer to a configuration which is either v-switching, or not univalent with respect to S_P, or not univalent with respect to $S_{P\cup\{p\}}$, for some process p.*

PROOF. Let $\alpha = \sigma_1\sigma_2\ldots$; note that $C' = (C, \vec{y}_1, \sigma_1(\vec{y}_1))$. Denote $C_0 = C$, $C_1 = C'$ and for every $k \geq 2$ fix $\vec{y}_k \in X^{C_{k-1}}$ and let $C_k = (C_{k-1}, \vec{y}_k, \sigma_k(\vec{y}_k))$.

Assume that for every $k \geq 1$, C_k is univalent, and let C_ℓ be the first configuration which is v-valent with respect to S_P. If such a configuration does not exist then the execution either reaches a configuration that decides $\bar{v}$, or does not reach any decision. Since α is v-deciding from C, the probability that C_ℓ does not exist is at most $\epsilon_k \leq \frac{1}{n\sqrt{n}}$.

Since C_ℓ is the first configuration which is v-valent, $C_{\ell-1}$ is $\bar{v}$-valent. Therefore, there is a $\bar{v}$-deciding adversary $\beta = \rho_1, \rho_2, \ldots$ from $C_{\ell-1}$ in S_P. Consider the configuration $D = (C_{\ell-1}, \vec{y}_\ell, \rho_1(\vec{y}_\ell))$. If D is not $(k+\ell, S_P)$-univalent then we are done. If D is $\bar{v}$-valent, then since C_ℓ is v-valent, by Lemma 5, there exists a layer L such that $(C_{\ell-1}, \vec{y}_\ell, L)$ is either not $(k+\ell, S_P)$-univalent or not $(k+\ell, S_{P\cup\{p\}})$-univalent, for some process p, in which case we are also done. Otherwise D is v-valent, which implies that it is v-switching with respect to S_P from $C_{\ell-1}$ by $\vec{y}_\ell$ and β. □

4.3 One-Round Coin-Flipping Games

The remaining part of the lower bound proof deals with null-valent configurations, and it relies on results about one-round coin-flipping games. Formally, a *U-valued one-round coin flipping game of m players* is a function

$$g : \{X_1 \cup \perp\} \times \{X_2 \cup \perp\} \times \cdots \times \{X_m \cup \perp\} \longrightarrow \{1, 2, \ldots, U\},$$

where X_i, $1 \leq i \leq m$, is the i-th probability space.

A *t-hiding adversary* may hide at most t of the random choices in $X_1, \cdots, X_m$, by replacing them with a $\perp$. Formally, let $X = X_1 \times \cdots \times X_m$ be the product probability space. For every $\vec{y} \in X$, and $I \subseteq \{1, \cdots, m\}$, the vector reached when the adversary hides the coordinates of I is defined as follows:

$$\vec{y}_I(i) = \begin{cases} \vec{y}(i), & i \notin I \\ \perp, & i \in I. \end{cases}$$

For every possible outcome of the game $u \in \{1, \ldots, U\}$, define the set of all vectors in X for which no t-hiding adversary can force the outcome of the game to be u, as

$$W^u = \{\vec{y} \in X | g(\vec{y}_I) \neq u \text{ for every } I \subseteq \{1, \cdots, m\} \text{ such that } |I| \leq t\}.$$

We prove that there is high probability for hiding values in a way that forces one of the outcomes when $U = 3$, i.e., for some $u \in \{1, 2, 3\}$, $\Pr[\vec{y} \in W^u]$ is very small. The proof relies on an isoperimetric inequality, following a result of Schechtman [27].

The space (X, d) is a finite metric space where for every $\vec{x}$ and $\vec{y}$ in X, $d(\vec{x}, \vec{y})$ is the Hamming distance between $\vec{x}$ and $\vec{y}$ (the number of coordinates in which $\vec{x}$ and $\vec{y}$ differ). For $A \subseteq X$, $B(A, t)$ is the *ball of radius t around the set A*, i.e.,

$$B(A, t) = \{\vec{y} \in X | \text{ there is } \vec{z} \in A \text{ such that } d(\vec{y}, \vec{z}) \leq t\}.$$

LEMMA 9. *Let $X = X_1 \times \cdots \times X_m$ be a product probability space and $A \subseteq X$ such that $Pr[\vec{x} \in A] = c$. Let $\lambda_0 = \sqrt{2m \log \frac{2}{c}}$, then for $\ell \geq \lambda_0$,*

$$Pr[\vec{x} \in B(A, \ell)] \geq 1 - 2e^{-\frac{(\ell-\lambda_0)^2}{2m}}.$$

PROOF. Consider an element $\vec{x}$ as a random function $\vec{x} : D = \{1, \ldots, m\} \rightarrow X_1 \cup \cdots \cup X_m$ such that $\vec{x}(i) \in X_i$. Define a sequence of partial domains $\emptyset = D_0 \subset D_1 \subset \cdots \subset D_m = D$ such that $D_i = \{1, \ldots, i\}$. Let $f : X \rightarrow \mathbb{R}$ be a function that measures the distance of elements from the given subset $A \subseteq X$, i.e., $f(\vec{x}) = d(A, \vec{x})$.

Choose a random element of $\vec{w} \in X$ according to the given distribution. Define the sequence $Y_0, \ldots, Y_m$ by $Y_i = \mathrm{E}[f(\vec{x}) \mid \vec{x}|_{D_i} = \vec{w}]$. Specifically, $Y_0 = \mathrm{E}[f(\vec{x})]$ with probability 1, and $Y_m = f(\vec{w})$ with the probability of choosing $\vec{w}$. It is well known that $Y_0, \ldots, Y_m$ is a Doob martingale (see [18, Chapter VI]).

Notice that $X_1, ..., X_m$ are independent and therefore for every $i \in D$, the random variable $\vec{x}(i)$ is independent of other values of $\vec{x}$.

The function f satisfies the Lipschitz condition, i.e., for every $\vec{x}, \vec{y}$ that differ only in $D_i \setminus D_{i-1}$ for some i, we have $|f(\vec{x}) - f(\vec{y})| \leq 1$, by the triangle inequality of the Hamming metric d. This implies that the martingale $Y_0, \ldots, Y_m$ satisfies the martingale Lipschitz condition, i.e., $|Y_i - Y_{i-1}| \leq 1$ for every i, $0 < i \leq m$.

By Azuma's inequality we have that

$$\Pr[|f(\vec{x}) - \mathrm{E}[f(\vec{x})]| > \lambda] < 2e^{-\frac{\lambda^2}{2m}}.$$

We now claim that $\mathrm{E}[f(\vec{x})] \leq \lambda_0$. Assume the contrary, that $\mathrm{E}[f(\vec{x})] > \lambda_0$. Since $\lambda_0 = \sqrt{2m \log \frac{2}{c}}$, we have that $2e^{-\frac{\lambda_0^2}{2m}} = c$. For every $\vec{x} \in A$ we have $f(\vec{x}) = 0$, therefore

$$\Pr[|f(\vec{x}) - \mathrm{E}[f(\vec{x})]| > \lambda_0] \geq \Pr[f(\vec{x}) = 0] = c,$$

which contradicts Azuma's inequality.

Hence, for every $\ell \geq \lambda_0$ we have

$$\begin{aligned} \Pr[\vec{x} \notin B(A, \ell)] &= \Pr[f(\vec{x}) > \ell] \\ &\leq \Pr[|f(\vec{x}) - \mathrm{E}[f(\vec{x})]| > \ell - \lambda_0] < 2e^{-\frac{(\ell-\lambda_0)^2}{2m}}, \end{aligned}$$

which completes the proof. □

LEMMA 10. *In a 3-valued one-round coin-flipping game of m players, for $t = 6\sqrt{2m \log(m^3)}$ there exists a t-hiding adversary and $u \in \{1, 2, 3\}$, such that the adversary can force the outcome of the game to u with probability greater than $1 - \frac{1}{m^3}$.*

PROOF. Recall that for every $u \in \{1, 2, 3\}$, the set W^u is the set of all vectors in X for which no t-hiding adversary

can force the outcome of the game to be u. So our goal is to prove that $\Pr[\vec{y} \in W^u] < \frac{1}{m^3}$ for some $u \in \{1,2,3\}$.

Denote $B^u = B(W^u, \frac{t}{3})$.

Assume that $\Pr[\vec{y} \in W^u] \geq \frac{1}{m^3}$ for all $u \in \{1,2,3\}$. Clearly, $\bigcap_{u\in\{1,2,3\}} W^u = \emptyset$, otherwise the value of the game is undefined for that point. Moreover, we claim that $\bigcap_{u\in\{1,2,3\}} B^u$ is empty. Assume there exists $\vec{z} \in \bigcap_{u\in\{1,2,3\}} B^u$. For every $u \in \{1,2,3\}$, let $\vec{x}^u \in W^u$ and $I_u \subseteq \{1,\ldots,m\}$ a set of indices, such that $|I_u| \leq \frac{t}{3}$ and $\vec{z}_{I_u} = \vec{x}^u_{I_u}$. Let $I = \bigcup_{u\in\{1,2,3\}} I^u$. Since $\vec{x}^u \in W^u$, we have that $g(\vec{x}^u_I) \neq u$, and so $g(\vec{z}_I) \neq u$. This implies that $g(\vec{z}_I)$ is undefined, and therefore $\bigcap_{u\in\{1,2,3\}} B^u = \emptyset$.

We now apply Lemma 9, with $A = W^u$. Notice that the results of the local flips of each player are independent random variables. We have that $\Pr[\vec{y} \in W^u] \geq c$ where $c = \frac{1}{m^3}$, and therefore $\lambda_0 = \sqrt{2m\log(2m^3)}$. Hence, for $\ell = \frac{t}{3} = 2\sqrt{2m\log(2m^3)} = 2\lambda_0$, we have

$$\begin{aligned}\Pr[\vec{y} \in B^u] &\geq 1 - 2e^{-\frac{(t-\lambda_0)^2}{2m}} \\ &= 1 - 2e^{-\frac{2m\log(2m^3)}{2m}} \\ &= 1 - 2e^{-\log(2m^3)} = 1 - \frac{1}{m^3}.\end{aligned}$$

Since $\bigcap_{u\in\{1,2,3\}} B^u = \emptyset$ we have that $\Pr[\vec{y} \in B^1 \cap B^2] + \Pr[\vec{y} \in B^1 \cap B^3] + \Pr[\vec{y} \in B^2 \cap B^2] \leq \frac{1}{2} \cdot \sum_{u\in\{1,2,3\}} \Pr[\vec{y} \in B^u]$, which implies that

$$\begin{aligned}\Pr[\vec{y} \in \cup_{u\in\{1,2,3\}} B^u] &= \sum_{u\in\{1,2,3\}} \Pr[\vec{y} \in B^u] \\ &\quad - \sum_{u\neq u'\in\{1,2,3\}} \Pr[\vec{y} \in B^u \cap B^{u'}] \\ &\quad + \Pr[\vec{y} \in \cap_{u\in\{1,2,3\}} B^u] \\ &\geq \frac{1}{2} \cdot \sum_{u\in\{1,2,3\}} \Pr[\vec{y} \in B^u] \\ &\geq \frac{3}{2} \cdot (1 - \frac{1}{m^3}) \\ &> 1\end{aligned}$$

This contradiction implies that for some $u \in \{1,2,3\}$ we have $\Pr[\vec{y} \in W^u] < \frac{1}{m^3}$. □

4.4 Null-Valent Configurations

We use one-round coin-flipping games to show that, with high probability, a null-valent configuration C can be extended with one f-layer to a null-valent configuration. In order to achieve the above, we may need to hide up to $6\sqrt{2n\log(2n^3)}$ processes in the layer other than the processes in P, although they are not permanently failed. Therefore, we assume that $f \geq 12\sqrt{2n\log(2n^3)}$, and will always make sure that $|P| \leq \frac{f}{2}$. This will allow us to hide $\frac{f}{2} \geq 6\sqrt{2n\log(2n^3)}$ additional processes (not in P), in executions in S_P.

LEMMA 11. *If a configuration C reachable by an f-execution is (k, S_P)-null-valent, then with probability at least $1 - \frac{1}{(n-|P|)^3}$, there is an f-adversary σ_1 such that $C \circ \sigma_1$ is $(k+1, S_P)$-null-valent.*

PROOF. Let C be a (k, S_P)-null-valent configuration, and let L^F be an f-layer that is full with respect to S_P. For every $\vec{y}_1 \in X^C$, we classify the configurations $(C, \vec{y}_1, L^F)$ into three categories:

1. The configuration $(C, \vec{y}_1, L^F)$ is $(0, k+1, S_P)$-potent.
2. The configuration $(C, \vec{y}_1, L^F)$ is $(1, k+1, S_P)$-valent.
3. The configuration $(C, \vec{y}_1, L^F)$ is $(k+1, S_P)$-null-valent.

This can be considered as a 3-valued one-round coin-flipping game of m players, where $m = n - |P|$. This implies that $n - f \leq m \leq n$. By Lemma 10, we can hide $6\sqrt{2m\log(2m^3)}$ processes and force the resulting configuration into one of the above categories with probability at least $1 - \frac{1}{m^3}$. Hiding processes is done by choosing a partial layer $L_{\vec{y}_1}$ in S_P that does not contain any step by the hidden processes, but only a step of each process that is not hidden. We define an adversary σ_1 that for each $\vec{y}_1 \in X$, chooses the corresponding partial layer, i.e., $\sigma_1(\vec{y}_1) = L_{\vec{y}_1}$.

Our claim is that the category that can be forced is the third one. Assume, towards a contradiction, that the category that can be forced is the first one.

This implies that the probability over $\vec{y}_1 \in X$ that $(C, \vec{y}_1, L_{\vec{y}_1})$ is $(0, k+1, S_P)$-potent is at least $1 - \frac{1}{m^3}$. Therefore with probability at least $1 - \frac{1}{m^3}$, a $\vec{y}_1 \in X$ is chosen such that there exists a 0-deciding adversary α' from $(C, \vec{y}_1, L_{\vec{y}_1})$ for which:

$$\Pr[\text{decision from } (C, \vec{y}_1, L_{\vec{y}_1}) \text{ under } \alpha' \text{ is } 0] > 1 - \epsilon_{k+1}$$

Therefore with probability at least $1 - \frac{1}{m^3}$, there exists an adversary $\alpha = \sigma_1, \alpha'$ from C such that:

$$\begin{aligned}&\Pr[\text{decision from } C \text{ under } \alpha \text{ is } 0] \\ &= \sum_{\vec{y}_1\in X} P(\vec{y}_1) \cdot \Pr[\text{decision from } (C, \vec{y}_1, L_{\vec{y}_1}) \text{ under } \alpha' \text{ is } 0] \\ &> \left(1 - \frac{1}{m^3}\right) \cdot (1 - \epsilon_{k+1}) \\ &\geq \left(1 - \frac{1}{(n-f)^3}\right) \cdot \left(1 - \frac{1}{n\sqrt{n}} + \frac{k+1}{(n-f)^3}\right) \\ &= 1 - \frac{1}{n\sqrt{n}} + \frac{k}{(n-f)^3} + \frac{1}{(n-f)^3 n\sqrt{n}} - \frac{k+1}{(n-f)^6} \\ &> 1 - \frac{1}{n\sqrt{n}} + \frac{k}{(n-f)^3} = 1 - \epsilon_k,\end{aligned}$$

where the last inequality holds for sufficiently large n, since $(n-f)^6 \geq (n-f)^3 n\sqrt{n}$ and $k = O(n)$.

This contradicts the assumption that C is (k, S_P)-null-valent. A similar argument holds for the second category.

Hence, with probability at least $1 - \frac{1}{m^3}$, the third category can be forced, namely, we can reach a configuration that is $(k+1, S_P)$-null-valent. □

4.5 Putting the Pieces Together

We can now put the pieces together and prove the lower bound on the total step complexity of any randomized consensus algorithm.

THEOREM 12. *The total step complexity of any f-tolerant randomized consensus algorithm in an asynchronous system, where $n-f \in \Omega(n)$ and $f \geq 12\sqrt{2n\log(2n^3)}$, is $\Omega(f(n-f))$.*

PROOF. We show that the probability of forcing the algorithm to continue $\frac{f}{2}$ layers is at least $1 - \frac{1}{\sqrt{n}}$. Therefore

the expected number of layers is at least $(1-\frac{1}{\sqrt{n}})\cdot\frac{f}{2}$. Each of these layers is an f-layer containing at least $n-f$ steps, implying that the expected total number of steps is at least $\Omega((1-\frac{1}{\sqrt{n}})\cdot\frac{f}{2}\cdot(n-f))$, which is in $\Omega(f(n-f))$ since $n-f\in\Omega(n)$.

We argue that for every k, $0\leq k\leq\frac{f}{2}$, with probability at least $1-k\frac{1}{n\sqrt{n}}$, there is a configuration C reachable by an f-execution with at least k layers, which is either v-switching or non-univalent with respect to S_P where $|P|\leq k+1$. Once the claim is proved, the theorem follows by taking $k=\frac{f}{2}$, since the probability of having an f-execution with more than $\frac{f}{2}$ layers is at least $1-\frac{f}{2}\frac{1}{n\sqrt{n}}>1-\frac{1}{\sqrt{n}}$.

We prove the claim by induction on k.

Basis: $k=0$. By Lemma 6, there exists an initial configuration C that is not univalent with respect to $S_\emptyset$ or $S_{\{p\}}$, for some process p.

Induction step: Assume C is a configuration reachable by an f-execution with at least k layers, that is either v-switching or non-univalent with respect to S_P where $|P|\leq k+1$. We prove that with probability at least $1-\frac{1}{n\sqrt{n}}$, C can be extended with at least one layer to a configuration C' that is either v-switching or non-univalent with respect to $S_{P'}$ where $|P'|\leq k+2$. This implies that C' exists with probability $(1-k\frac{1}{n\sqrt{n}})(1-\frac{1}{n\sqrt{n}})\geq 1-(k+1)\frac{1}{n\sqrt{n}}$.

If C is bivalent, then by Lemma 7, there exists an adversary σ and a process p such that $C\circ\sigma$ is either v-switching or not $(k+1,S_P)$-univalent or not $(k+1,S_{P\cup\{p\}})$-univalent.

If C is v-switching, then by Lemma 8, there exists a finite adversary α_ℓ and a process p such that with probability at least $1-\frac{1}{n\sqrt{n}}$, $C\circ\alpha_\ell$ is either $\bar{v}$-switching, or not univalent with respect to S_P, or not univalent with respect to $S_{P\cup\{p\}}$.

If C is null-valent, then by Lemma 11, there exists an adversary σ_1 such that the configuration $C\circ\sigma_1$ is not $(k+1,S_P)$-univalent with probability at least $1-\frac{1}{m^3}$. Since $m\geq n-f\in\Omega(n)$, we have that $1-\frac{1}{m^3}\geq 1-\frac{1}{(n-f)^3}>1-\frac{1}{n\sqrt{n}}$. □

Finally, taking $f\in\Omega(n)$ and $n-f\in\Omega(n)$, we get a lower bound of $\Omega(n^2)$ on the total step complexity.

5. A MATCHING UPPER BOUND

This section presents a randomized consensus algorithm with $O(n^2)$ total step complexity, by introducing a *shared coin* algorithm with a constant *agreement parameter* and $O(n^2)$ total step complexity. In a shared coin algorithm with agreement parameter δ, each process outputs a decision value -1 or $+1$, such that for every $v\in\{-1,+1\}$, there is a probability of at least δ for all processes to output the same value v [5]. Using a shared coin algorithm with $O(n^2)$ total step complexity and a constant agreement parameter, in the framework of Aspnes and Herlihy [5], implies a randomized consensus algorithm with $O(n^2)$ total step complexity.

As in previous shared coin algorithms [13, 26], in our algorithm the processes flip coins until the amount of coins that were flipped reaches a certain threshold. An array of n single-writer multi-reader registers records the number of coins each process has flipped, and their sum. A process reads the whole array in order to track the total number of coins that were flipped.

Each process decides on the value of the majority of the coin flips it reads. Our goal is for the processes to read similar sets of coins, in order to agree on the same majority value.

Algorithm 1 Shared coin algorithm: code for process p_i.

```
local integer num, initially 0
1: while not done do
2:     num++
3:     flip = random(−1,+1)                    // a fair local coin
4:     A[i].⟨count, sum⟩ = ⟨count++, sum + flip⟩
                                               // atomically
5:     if num = 0 mod n then   // check if time to terminate
6:         collect A                           // n read operations
7:         if A[1].count + ... + A[n].count > n^2 then
               done = true                     // raise termination flag
   end while
8: collect A                                   // n read operations
9: return sign(∑_{j=1}^{n} A[j].sum)           // return +1 or −1,
       // depending on the majority value of the coin flips
```

For this to happen, we bound the total number of coins that are flipped (by any process) after some process observes that the threshold was exceeded. A very simple way to guarantee this property is to have processes frequently read the array in order to detect quickly that the threshold was reached. This, however, increases the total step complexity.

The novel idea of our algorithm in order to overcome this conflict, is to utilize a multi-writer register called *done* that serves as a binary termination flag; it is initialized to false. A process that detects that enough coins were flipped, sets *done* to true. This allows a process to read the array only once in every n of its local coin flips, but check the register *done* before each local coin flip.

The pseudocode appears in Algorithm 1. In addition to the binary register *done*, it uses an array A of n single-writer multi-reader registers, each with the following components (all initially 0):

***count*:** how many flips the process performed so far.

***sum*:** the sum of coin flips values so far.

For the proof, fix an execution α of the algorithm. We will show that all processes that terminate agree on the value 1 for the shared coin with constant probability; by symmetry, the same probability holds for agreeing on -1, which implies the algorithm has a constant agreement parameter.

The total count of a specific collect is the sum of $A[1].count,\ldots,A[n].count$, as read in this collect. Note that the total count in Line 8 is ignored, but it can still be used for the purpose of the proof.

Although we only maintain the counts and sums of coin flips, we can (externally) associate them with the set of coin flips they reflect; we denote by F_C the collection of coin flips that are written in the shared memory by the first time that true is written to *done*. The size of F_C can easily be bounded, since each process flips at most n coins before checking A.

LEMMA 13. *F_C contains at least n^2 coins and at most $2n^2$ coins.*

For a set of coins F we let $Sum(F)$ be the sum of the coins in F. We denote by F_i the set of coin flips read by the collect of process p_i in Line 8. This is the set according to which the process p_i decides on its output, i.e., p_i returns

$Sum(F_i)$. Since each process may flip at most one more coin after true is written to *done*, we can show:

LEMMA 14. *For every i, $F_C \subseteq F_i$, and $F_i \setminus F_C$ contains at most $n-1$ coins.*

We now show that there is at least a constant probability that $Sum(F_C) \geq n$. In this case, by Lemma 14 all processes that terminate agree on the value 1, since F_i contains at most $n-1$ additional coins.

We partition the execution into three phases. The first phase ends when n^2 coins are written. We assume a stronger adversary that can choose these n^2 coins out of n^2+n-1 coins that were flipped. The second phase ends when true is written to *done*. In the third phase, each process reads the whole array A and returns a value for the shared coin.

Since F_C is the set of coins written when *done* is set to true, then it is exactly the set of coins written in the first and second phases. Let F_{first} be the first n^2 coins that are written, and $F_{second} = F_C \setminus F_{first}$. (See Figure 2.) This implies that $Sum(F_C) = Sum(F_{first}) + Sum(F_{second})$. Therefore, we can bound (from below) the probability that $Sum(F_C) \geq n$ by bounding the probabilities that $Sum(F_{first}) \geq 3n$ and $Sum(F_{second}) \geq -2n$.

Consider the sum of the first n^2+n-1 coin flips. After these coins are flipped, the adversary has to write at least n^2 of them, which will be the coins in F_{first}. If the sum of the first n^2+n-1 coin flips is at least $4n$ then $Sum(F_{first}) \geq 3n$. We bound the probability that this happens using the Central Limit Theorem.

LEMMA 15. *The probability that $Sum(F_{first}) \geq 3n$ is at least $\frac{1}{8\sqrt{2\pi}}e^{-8}$.*

PROOF. There are $N = n^2+n-1$ coins flipped when n^2 coins are written to F_{first}. By the Central Limit Theorem, the probability for the sum of these coins to be at least $x\sqrt{N}$, converges to $1-\Phi(x)$, where $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{1}{2}y^2}dy$ is the normal distribution function. By Feller [17, Chapter VII], we have $1-\Phi(x) > (\frac{1}{x}-\frac{1}{x^3})\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$. Substituting $x=4$ we have that with probability at least $\frac{1}{8\sqrt{2\pi}}e^{-8}$ the sum of these N coins is at least $4\sqrt{N}$, which is more than $4n$, and hence $Sum(F_{first}) \geq 3n$. □

We now need to bound $Sum(F_{second})$. Unlike F_{first}, whose size is determined, the adversary may have control over the number of coins in F_{second}, and not only over which coins are in it. However, by Lemma 13 we have $|F_C| \leq 2n^2$, therefore $|F_{second}| \leq n^2$, which implies that F_{second} must be some prefix of n^2 additional coin flips. We consider the partial sums of these n^2 additional coin flips, and show that with high probability, all these partial sums are greater than $-2n$, and therefore in particular $Sum(F_{second}) > -2n$.

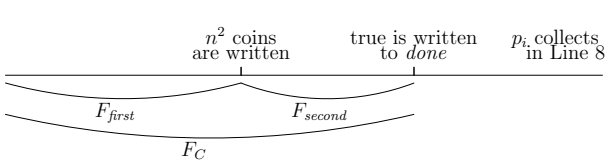

Figure 2: Phases of the shared coin algorithm.

Formally, for every i, $1 \leq i \leq n^2$, let X_i be the i-th additional coin flip, and denote $S_j = \sum_{i=1}^{j} X_i$. Since $|F_{second}| \leq n^2$, there exist k, $1 \leq k \leq n^2$, such that $S_k = Sum(F_{second})$. If $S_j > -2n$ for every j, $1 \leq j \leq n^2$, then specifically $Sum(F_{second}) = S_k > -2n$.

The bound on the partial sums is derived using Kolmogorov's inequality.

LEMMA 16. *The probability that $S_j > -2n$ for all j, $1 \leq j \leq n^2$, is at least $\frac{3}{4}$.*

PROOF. The results of the n^2 coin flips are independent random variables $X_1, \ldots X_{n^2}$, with $E[X_i] = 0$ and $Var[X_i] = 1$, for every i, $1 \leq i \leq n^2$. Since S_j is the sum of j independent random variables, its expectation is $E[S_j] = \sum_{i=1}^{j} E[X_i] = 0$, and its variance is $Var[S_j] = \sum_{i=1}^{j} Var[X_i] = j$. Kolmogorov's inequality [17, Chapter IX] implies that $|S_j| < 2n$, for all j, $1 \leq j \leq n^2$, with probability at least $\frac{3}{4}$. □

This bounds the probability of agreeing on the same value for the shared coin as follows.

LEMMA 17. *Algorithm 1 is a shared coin algorithm with agreement parameter $\delta = \frac{3}{32\sqrt{2\pi}}e^{-8}$.*

PROOF. By Lemma 15, the probability that $Sum(F_{first}) \geq 3n$ is at least $\frac{1}{8\sqrt{2\pi}}e^{-8}$, and by Lemma 16, the probability that $Sum(F_{second}) \geq -2n$ is at least $\frac{3}{4}$. Since $Sum(F_C) = Sum(F_{first}) + Sum(F_{second})$, this implies that the probability that $Sum(F_C) \geq n$ is at least $\frac{3}{32\sqrt{2\pi}}e^{-8}$.

By Lemma 14, for every i, $F_i \setminus F_C$ contains at most $n-1$ coins, which implies that if $Sum(F_C) \geq n$ then $Sum(F_i) \geq 1$, and therefore if p_i terminates, then it will decide 1. Hence with probability at least $\frac{3}{32\sqrt{2\pi}}e^{-8}$, $Sum(F_C) \geq n$ and all processes which terminate agree on the value 1.

By symmetry, all processes which terminate agree on the value -1 with at least the same probability. □

Clearly, Algorithm 1 flips $O(n^2)$ coins. Moreover, all work performed by processes in reading the array A can be attributed to coin flips. This can be used to show that Algorithm 1 has $O(n^2)$ total step complexity.

Using Algorithm 1 in the framework of Aspnes and Herlihy [5] implies:

THEOREM 18. *There is a randomized consensus algorithm with $O(n^2)$ total step complexity.*

6. DISCUSSION

We have proved that $\Theta(n^2)$ is a tight bound on the total step complexity of solving randomized consensus, under a strong adversary, in an asynchronous shared-memory system using multi-writer registers.

Aspnes [3] shows an $\Omega(\frac{n^2}{\log^2 n})$ lower bound on the expected total number of coin flips. Our layering approach as presented here, does not distinguish between a deterministic step and a step involving a coin flip, leaving open the question of the optimal number of coin flips.

Our algorithm exploits the multi-writing capabilities of the register. The best known randomized consensus algorithm using single-writer registers [13] has $O(n^2 \log n)$ total

step complexity, and it is intriguing to close the gap from our lower bound.

In addition to settling the open question of the asymptotic total step complexity, we consider an important contribution of this paper to be in introducing new techniques for investigating randomized consensus algorithms. These techniques provide a new context and opportunities for exploring several research directions, two of which are discussed next.

One direction is to study the *individual* step complexity of randomized consensus; namely, the expected number of steps performed by a single process. Aspnes and Waarts [6] present a shared coin algorithm in which each process performs $O(n \log^2 n)$ expected number of steps; their algorithm has $O(n^2 \log^2 n)$ total step complexity. Their shared coin algorithm is similar to [13], but the local flips of a process are assigned increasing weights, which allows fast processes to terminate earlier. We believe that the individual step complexity can be reduced using multi-writer registers.

Our results may also provide new insight into the expected step complexity of randomized consensus under *weak* adversaries that cannot observe the local flips or the contents of the shared memory. Several papers presented randomized consensus algorithms for this type of adversaries, e.g., [9–11, 14, 15], with the best algorithms having $O(n \text{ polylog}(n))$ total step complexity and $O(\text{polylog}(n))$ individual step complexity. It will be intriguing to find tight bounds for this model as well.

Acknowledgements. We would like to thank Faith Ellen, Javier Garcia, David Hay, Danny Hendler, Sergio Rajsbaum and an anonymous referee for insightful comments on earlier versions of the paper.

7. REFERENCES

[1] K. Abrahamson. On achieving consensus using a shared memory. *PODC* 1988, pp. 291–302.

[2] J. Aspnes. Time- and space-efficient randomized consensus. *PODC* 1990, pp. 325–332.

[3] J. Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. *J. ACM*, 45(3):415–450, May 1998.

[4] J. Aspnes. Randomized protocols for aynchronous consensus. *Dist. Comp.*, 16(2-3):165–176, Sept. 2003.

[5] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *J. Alg.*, 11(3):441–461, 1990.

[6] J. Aspnes and O. Waarts. Randomized consensus in expected $O(n \log^2 n)$ operations per processor. *SIAM J. Comput.*, 25(5):1024–1044, 1996.

[7] H. Attiya, D. Dolev, and N. Shavit. Bounded polynomial randomized consensus. *PODC* 1989, pp. 281–293.

[8] H. Attiya and J. L. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics.* John Wiley & Sons, 2nd edition, 2004.

[9] Y. Aumann. Efficient asynchronous consensus with the weak adversary scheduler. *PODC* 1997, pp. 209–218.

[10] Y. Aumann and M. A. Bender. Efficient low-contention asynchronous consensus with the value-oblivious adversary scheduler. *Dist. Comp.*, 17(3):191–207, Mar. 2005.

[11] Y. Aumann and A. Kapah-Levy. Cooperative sharing and asynchronous consensus using single-reader single-writer registers. *SODA* 1999, pp. 61–70.

[12] Z. Bar-Joseph and M. Ben-Or. A tight lower bound for randomized synchronous consensus. *PODC* 1998, pp. 193–199.

[13] G. Bracha and O. Rachman. Randomized consensus in expected $O(n^2 \log n)$ operations. *WDAG* 1991, pp. 143–150.

[14] T. D. Chandra. Polylog randomized wait-free consensus. *PODC* 1996, pp. 166–175.

[15] B. Chor, A. Israeli, and M. Li. On processor coordination using asynchronous hardware. *PODC* 1987, pp. 86–97.

[16] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[17] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, 3rd edition, 1968.

[18] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. John Wiley & Sons, 2nd edition, 1971.

[19] M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[20] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, Apr. 1985.

[21] M. Herlihy. Wait-free synchronization. *ACM Trans. Prog. Lang. Syst.*, 13(1):124–149, January 1991.

[22] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[23] M. C. Loui and H. H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, pp. 163–183, 1987.

[24] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.

[25] Y. Moses and S. Rajsbaum. The unified structure of consensus: A layered analysis approach. *PODC* 1998, pp. 123–132.

[26] M. Saks, N. Shavit, and H. Woll. Optimal time randomized consensus—making resilient algorithms fast in practice. *PODC* 1991, pp. 351–362.

[27] G. Schechtman. Levy type inequality for a class of finitie metric spaces. In *Lecture Notes in Mathematics: Martingle Theory in Harmonic Analysis and Banach Spaces*, volume 939, pp. 211–215. Springer-Verlag, 1981.

[28] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, Dec. 1990.