Lower Bounds for Distributed Coin-Flipping and Randomized Consensus

James Aspnes [*]

Abstract

We examine a class of *collective coin-flipping games* that arises from randomized distributed algorithms with halting failures. In these games, a sequence of *local coin flips* is generated, which must be combined to form a single *global coin flip*. An adversary monitors the game and may attempt to bias its outcome by hiding the result of up to t local coin flips. We show that to guarantee at most constant bias, $\Omega(t^2)$ local coins are needed, even if (a) the local coins can have arbitrary distributions and ranges, (b) the adversary is required to decide immediately whether to hide or reveal each local coin, and (c) the game can detect which local coins have been hidden. If the adversary is permitted to control the outcome of the coin except for cases whose probability is polynomial in t, $\Omega(t^2/\log^2 t)$ local coins are needed. Combining this fact with an extended version of the well-known Fischer-Lynch-Paterson impossibility proof of deterministic consensus, we show that given an adaptive adversary, any t-resilient asynchronous consensus protocol requires $\Omega(t^2/\log^2 t)$ local coin flips in any model that can be simulated deterministically using atomic registers. This gives the first non-trivial lower bound on the total work required by wait-free consensus and is tight to within logarithmic factors.

1 Introduction

Our results divide naturally into two parts: a lower bound for asynchronous randomized consensus in a wide variety of models, and a still more general lower bound for a large class of collective coin-flipping games that forms the basis of the consensus lower bound but is interesting in its own right.

Consensus is a fundamental problem in distributed computing in which a group of processes must agree on a bit despite the interference of an adversary. (An additional condition forbids trivial solutions that always produce the same answer). In an asynchronous setting, it has long been known that if an adversary can halt a single process, then no deterministic consensus algorithm is possible without the use of powerful synchronization primitives [CIL87, DDS87, FLP85, Her91, LAA87].

In contrast, randomized algorithms can solve consensus in a shared-memory system for n processes even if the adversary can halt up to $n-1$ processes. Such algorithms are called *wait-free* [Her91] because any process can finish the algorithm without waiting for slower (or possibly dead) processes. These algorithms work even under the assumption that failures and the timing of all events in the system are under the control of an *adaptive adversary*— one that can observe and react to all aspects of the system's execution (including the internal states of the processes).

The first known algorithm that solves shared-memory consensus against an adaptive adversary is the exponential-time algorithm of Abrahamson [Abr88]; since its appearance, numerous polynomial-time algorithms have appeared [AH90, ADS89, SSW91, Asp93, DHPW92, BR90, BR91, AW96]. Most of these algorithms are built around *shared coin protocols* in which the processes individually generate many random ± 1 *local coin flips*, which are combined by majority voting. The adversary may bias the outcome of the voting by selectively killing processes that have chosen to vote the "wrong" way before they can reveal their most recent votes to the other processes. To prevent the adversary from getting more than a constant bias, it is necessary to collect enough votes that the hidden votes shift the outcome by no more than a constant number of standard deviations. With up to $n-1$ failures (as in the wait-free case), this requires a total of $\Omega(n^2)$ local coin-flips, and at least $\Omega(n^2)$ work in order to communicate these coin-flips.[1]

[*]Yale University, Department of Computer Science, 51 Prospect Street/P.O. Box 208285, New Haven, CT 06520-8285. Supported by NSF grants CCR-9410228 and CCR-9415410. E-mail: aspnes@cs.yale.edu.

[1]Some of the algorithms deviate slightly from the simple

Improvements in other aspects of consensus algorithms have steadily brought their costs down, from the $O(n^4)$ total work of [AH90] to the $O(n^2 \log n)$ total work of [BR91]. But while these algorithms have steadily approached the $\Omega(n^2)$ barrier, none have broken it. However, no proof was known that consensus could not be solved in less than $\Omega(n^2)$ time; the barrier was solely a result of the apparent absence of alternatives to using shared coins based on majority voting. Indeed, it was asked in [Asp93] if it was necessarily the case that (a) every consensus protocol contained an embedded shared coin protocol; and (b) no shared coin protocol could achieve better performance than majority voting.

1.1 Our Results

In this paper we answer both of these questions, though the answers are not as simple as might have been hoped. We show that (a) every t-resilient asynchronous consensus protocol in the shared-memory model, with at least constant probability, either executes a shared coin protocol with bias at most one minus a polynomial in t or carries out an expected $\Omega(t^2)$ local coin-flips avoiding it; and (b) any such shared coin protocol requires an expected $\Omega(t^2/\log^2 t)$ local coin flips. It follows that t-resilient asynchronous consensus requires an expected $\Omega(t^2/\log^2 t)$ local coin flips. Since protocols based on majority voting require only $O(t^2)$ local coin flips, this lower bound is very close to being tight.

Since we are counting coin-flips rather than operations, the lower bound is not affected by deterministic simulations. So, for example, it continues to hold in message-passing models with up to t process failures (since a message channel can be simulated by an unboundedly large register), or in a shared-memory model with counters or cheap atomic snapshots. Furthermore, since our lower bound assumes that local coin flips can have arbitrary ranges and distributions, we may assume without loss of generality that any two successive coin-flips by the same process are separated by at least one deterministic operation in any of these models— so the lower bound on local coin-flips in fact implies a lower bound on total work.

The lower bound on coin-flipping games is still more general, and holds in any model in which the adversary may intercept up to t local coin-flips before they are revealed, no matter what (deterministic) synchronization primitives or shared objects are available. Furthermore, it is tight in the sense that it shows that no *constant-bias* shared coin can use less than $\Omega(t^2)$ local coins, a bound achieved (ignoring constants) by majority voting.

majority-voting approach described here. In the algorithm of Aspnes [Asp93], some votes are generated deterministically. In the algorithm of Saks, Shavit, and Woll [SSW91], several coin-flipping protocols optimized for different execution patterns are run in parallel. In the algorithm of Aspnes and Waarts [AW96], processes that have already cast many votes generate votes with increasing weights in order to finish the protocol quickly. However, none of these protocols costs less than simple majority voting in terms of the expected total number of local coin flips performed in the worst case.

1.2 Related Work

Many varieties of collective coin-flipping games have been studied, starting with the work of Ben-Or and Linial [BOL85]. Many such games assume that the locations of faulty coins are fixed in advance; under these assumptions very efficient games exist [AN90, CL93, BOL85, Sak89]. Another assumption that greatly limits the power of the adversary is to require that both the locations and values of faulty coins are fixed in advance; this is the *bit extraction problem* [CFG+85, Fri92, Vaz85], in which it is possible to derive completely unbiased random bits.

If none of these limiting assumptions are made, the adversary gains considerably more power. An excellent survey of results for a wide variety of models involving fair or nearly fair two-valued local coins can be found in [BOLS87]. Our work differs from these in that we allow arbitrary distributions on the local coins. With a restriction to fair coins, Harper's isoperimetric inequality for the hypercube [Har66] implies that the majority function gives the least power to an off-line adversary that can see all coins before deciding which to change; and Lichtenstein, Linial, and Saks [LLS89] have shown that majority is also optimal against an on-line adversary similar to the one we consider here. Without such a restriction, the best previously known bound is a bound of $\Omega(1/\sqrt{n})$ on the influence of an adversary that can hide *one* coin; this is an easy corollary of a theorem about gaps in martingale sequences due to Cleve and Impagliazzo [CI93].

A very nice lower bound on the *space* used by wait-free shared-memory consensus is due to Fich, Herlihy, and Shavit [FHS93]. They show that any such consensus protocol must use $\Omega(\sqrt{n})$ distinct registers to guarantee agreement. Unfortunately, their techniques do not appear to generalize to showing lower bounds on work.

2 Coin-Flipping Games

A collective coin-flipping game [BOL85] is an algorithm for combining many *local coins* into a single *global coin*, whose bias should be small even though some of the local coins may be obscured by a malicious adversary. Though the particular coin-flipping games we consider here are motivated by their application to proving lower bounds on distributed algorithms with failures, they abstract away almost all of the details of the original distributed systems and are thus likely to be useful in other contexts.

We assume that the local coins are independent random variables whose ranges and distributions are arbitrary. The values of these variables are revealed one at a time to an adversary who must immediately choose whether to reveal or obscure each value. If the adversary chooses to obscure the value of a particular local coin, the effect is to replace it with a default value $\perp$. Repeating this process yields a sequence of values, some of which are the original values of the random variable and some of which are $\perp$. A function is applied to this sequence to yield an *outcome*, which may be arbitrary but which we will usually require to be ± 1. The adversary's power is limited by an upper bound on how many coins it may obscure.

Note that in this description we assume that the adversary cannot predict future local coins; it can only base its decision to reveal or obscure a particular coin on its value and the values of earlier coins. In addition, the adversary's interventions are visible. The coin-flipping game may observe and react to the fact that the adversary has chosen to obscure particular local coins, even though it has no access to the true values of those coins.

Formally, a coin-flipping game is specified by a tree. The leaves of the tree specify the outcomes of the game. Internal nodes correspond to local coin-flips. Coin-flipping games are defined recursively as follows. Fix a set of possible outcomes. A coin-flipping game G with maximum length zero consists of a single outcome; we will call such a game a *constant game* and abuse notation by writing its outcome simply as G. A coin-flipping game G with maximum length n is either a constant game or consists of

1. A random variable representing the first local coin-flip in G.
2. A function mapping the range of this random variable to the set of coin-flipping games with maximum length less than n (the *subgames* of G). For each value α in this range, the resulting subgame is denoted G_α.
3. A *default subgame* $G_\perp$ with maximum length less than n, corresponding to the effect of an adversary choice to hide the first local coin-flip in G.

The above definition represents a coin-flipping game as a tree; if we think of G as the root of the tree its children are the subgames G_α for each value of α and the default subgame $G_\perp$. The actual game tree corresponding to playing the game against an adversary is a bit more complicated and involves two plies for each level of G. We may think of the states of this game as pairs (G, k) specifying the current subgame G and the limit k on how many local coins the adversary may hide (i.e., the number of *faults*). To execute the first local coin-flip in G, two steps occur. First, the outcome α of the coin-flip is determined. Second, the adversary chooses between revealing α, leading to the state (G_α, k); or hiding α, leading to the state $(G_\perp, k-1)$.

In order to prevent the adversary from being able to predict the future or the game from being able to deduce information about obscured coins, we demand that all random variables on any path through the game tree be independent.

An adversary strategy specifies for each partial sequence of local coin-flips whether to hide or reveal the last coin. We will write $G \circ A$ for the random variable describing the outcome of G when run under the control of an adversary strategy A. If a game G has real-valued outcomes, then for each number of faults k there exist adversary strategies to maximize or minimize the expected outcome. Define $M_k G$ to be the maximum expected outcome and $m_k G$ to be the minimum expected outcome. These values can be computed recursively as follows:

- If G has length 0, $M_k G = m_k G = G$.
- If G has positive length, then

$$\begin{aligned} M_k(G) &= \mathrm{E}_\alpha\left[\max\left(M_k G_\alpha, M_{k-1} G_\perp\right)\right] && (1) \\ m_k(G) &= \mathrm{E}_\alpha\left[\min\left(m_k G_\alpha, m_{k-1} G_\perp\right)\right]. && (2) \end{aligned}$$

Most of the time we will assume that the only possible outcomes of a game are ± 1. In this case the quantities M_k and m_k give a measure of how much influence an adversary with the ability to hide k local coin-flips can get over the outcome. It is necessary to consider both at once: as we will see later, it is always possible to find a game with maximum length n whose minimum expected outcome m_k can be any value in the range $[-1, 1]$. We will be interested in the best such game, i.e., the one that attains a particular value of m_k while minimizing M_k (or, symmetrically, the game that maximizes m_k for a particular fixed M_k). In general it will turn out to be quite difficult to find this game exactly (although much can be shown about its structure), and so it will be necessary to settle for a lower bound on $M_k G$ as a function of n, k, and $m_k G$.

2.1 The Structure of Optimal Games

Fix a maximum length n and number of failures k. Let us define the *range* of a game G to be the interval $[m_k G, M_k G]$. Then G *(strictly) dominates* G' just in case the range of G is a (proper) subset of the range of G'; in other words, if G gives the adversary no more control than G' does. A game G is *optimal* if it either dominates all other games G' with $m_k G' = m_k G$ or if it dominates all other games G' with $M_k G' = M_k G$. For $k < n$, this definition will turn out to be equivalent to saying that no game strictly dominates G.

With each k and game G we can associate a point in a two-dimensional space given by the coordinates $m_k G$ and $M_k G$. From this geometric perspective the problem we are interested in is finding for each value of n and k the curve corresponding to the set of optimal games with maximum length n and up to k failures.

For some values of n and k this task is an easy one. If $k = 0$, then the $(n, 0)$ curve is just the diagonal running from $(-1, -1)$ to $(1, 1)$, since $m_0 G = M_0 G$ for all G. If the other extreme holds and $k \geq n$, then for any G either $m_k G = -1$ or $M_k G = 1$, depending on the default outcome of G if all local coins are hidden. It is not difficult to see that if $M_n G = 1$, then $m_n G$ can be any value between -1 and 1. For example, G could set its outcome to be the value of the first local coin, or 1 if that coin-flip is hidden; if the adversary wishes to achieve an outcome lower than 1 it must let the first local coin go through. Similar, if $m_n G = -1$ then $M_n G$ can be any value between -1 and 1. Thus the optimal (n, n) curve consists of the line segment from $(-1, -1)$ to $(-1, 1)$ and the line segment from $(-1, 1)$ to $(1, 1)$.

Equations (1) and (2) have a nice geometrical interpretation that in principle allows one to determine the (n, k) curves of optimal games of maximum length n with k failures. This process is depicted in Figures 1 and 2. Fix a game G. Each subgame G_α corresponds to a point $(m_k G_\alpha, M_k G_\alpha)$, which must lie somewhere on or above the curve of optimal $(n-1, k)$ games. The contribution of G_α to the position of G is given by $(\min(m_k G_\alpha, m_{k-1} G_\perp), \max(M_k G_\alpha, M_{k-1} G_\perp))$, which is a point in the intersection of the region above the $(n-1, k)$ curve and the rectangle of points dominated by $G_\perp$. Since the value of G is the average of these contributions, it must correspond to some point in the convex closure of this intersection. Provided the $(n-1, k)$ curve is concave (which is easily proved by induction on $n-k$ as shown below), then all points in the convex closure are dominated by some point on its lower right edge: the line segment between the optimal (n, k) game G_0 with $M_k G_0 = M_{k-1} G_\perp$ and the optimal (n, k) game G_1 with $m_k G_1 = m_{k-1} G_\perp$.

Geometrically, this edge is the hypotenuse of a right triangle inscribed between the $(n-1, k)$ and $(n-1, k-1)$ curves such that its sides are parallel to the axes and its right corner is on the $(n-1, k-1)$ curve. To take into account all possible choices of $G_\perp$, it is necessary to consider all such triangles. By taking the minimum of the hypotenuses of these triangles (as shown in Figure 2), we obtain the (n, k) curve of all optimal games of maximum length n subject to up to k failures. Note that if the $(n-1, k)$ curve is nondecreasing and concave (true for $n-1 = k$, true as the induction hypothesis for larger $n-1$), we may extend each hypotenuse to its containing line without affecting the minimum, and so the (n, k) curve as the minimum of concave functions is also nondecreasing and concave.

Let us summarize. From the discussion of the constraints on G given $G_\perp$, we have:

Theorem 1 *For each coin-flipping game G with maximum length n and up to k failures, there is a G' such that G' dominates G, $G'_\perp$ dominates $G_\perp$, and G' has two non-default subgames G'_0 and G'_1 with $M_k G'_0 = M_{k-1} G'_\perp$ and $m_k G'_1 = m_{k-1} G'_\perp$.*

One consequence of this theorem is that we can replace any optimal G with an equivalent G' in which the first local coin has exactly two outcomes, and the adversary never prefers hiding a local coin to revealing one. Since the theorem also applies recursively to all subgames of G, we may assume that these conditions in fact hold throughout G'. Thus no additional power is obtained by allowing more than two outcomes to a coin. However, the theorem does not imply that we can require that all local coins are fair; indeed, for most optimal games they will not be.

In addition, we have shown the following about the shape of the curves corresponding to optimal games:

Theorem 2 *Fix n and k with $k < n$. For each x in $[-1, 1]$, let $f(x)$ be the smallest value of $M_k G$ for all G such that $m_k G = x$. Then f is nondecreasing and concave.*

Unfortunately, with the exception of some extreme cases like $k = n-1$, the (n, k) curves do not appear to have nice algebraic descriptions. So while in principle equations (1) and (2) and the minimum-of-hypotenuses construction constrain the curves completely, to obtain any useful bounds from them we will be forced to resort to approximation.

2.2 Lower Bounds on Coin-Flipping Games

The essential idea of our lower bound for coin-flipping games is to choose a family of functions to act as lower bounds for the optimal curves as defined above, and show by repeating the inscribed-right-triangle argument with these functions that they do in fact provide lower bounds on the optimal curves given appropriate parameters. The particular family of functions that we use consists of all hyperbolas that are symmetric about the diagonal from $(-1, 1)$ to $(1, -1)$ and that pass through the corner points $(-1, -1)$ and $(1, 1)$.[2] These hyperbolas are conveniently given by

$$\tanh^{-1} y - \tanh^{-1} x = c$$

for various values of c. The linear $(n, 0)$ curve corresponds exactly to $c = 0$; the (n, n) curve is the limit as c goes to infinity. Our goal is to compute values of c as a function of n and k such that for all length-n games,

$$\tanh^{-1} M_k G - \tanh^{-1} m_k G \geq c(n, k).$$

Given $c(n-1, k)$ and $c(n-1, k-1)$, repeating the inscribed-right-triangle construction for the resulting hyperbolas is a not very difficult exercise in analytic geometry. Unfortunately, finding the particular point on the hypotenuse of the particular triangle that minimizes $c(n, k)$ is a bit more involved (details of both steps are given in the full paper). The ultimate result of these efforts is:

[2]We conjecture that a slightly tighter lower bound could be proven using the curves given by $\Phi^{-1}(y) - \Phi^{-1}(x) = c$, where Φ is the normal distribution function. An analog of Theorem 3 using Φ instead of tanh would improve the consensus lower bound in Theorem 6 by a logarithmic factor.

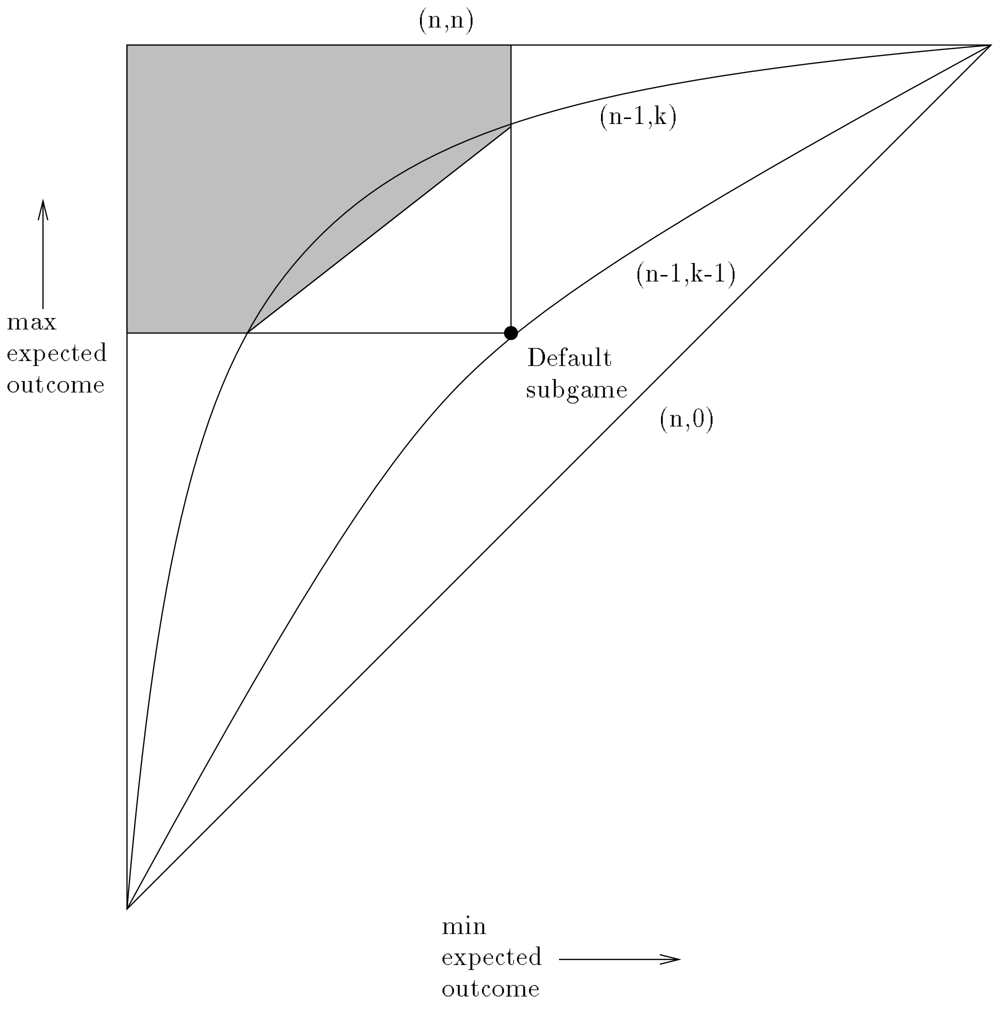

Figure 1: Graphical depiction of constraints on minimum and maximum expected outcomes of a game G given n and k. Each point in the figure corresponds to a pair of minimum and maximum expected outcomes. The diagonal represents the $k = 0$ case where these values are the same. The outer edges of the figure represent the $k = n$ case. The two inner curves represent all optimal games with $n-1$ voters and either k or $k-1$ failures. The default subgame $G_\perp$ lies somewhere on or above the $(n-1, k-1)$ curve. All other subgames G_α lie on or above the $(n-1, k)$ curve. If $G_\perp$ is fixed, the value of G lies somewhere in the convex closure of the intersection of the region above the $(n-1, k)$ curve and the rectangle dominated by $G_\perp$. All points in this convex closure, shown shaded in the picture, are dominated by some point on the hypotenuse of the right triangle inscribed between the $(n-1, k)$ and $(n-1, k-1)$ curves.

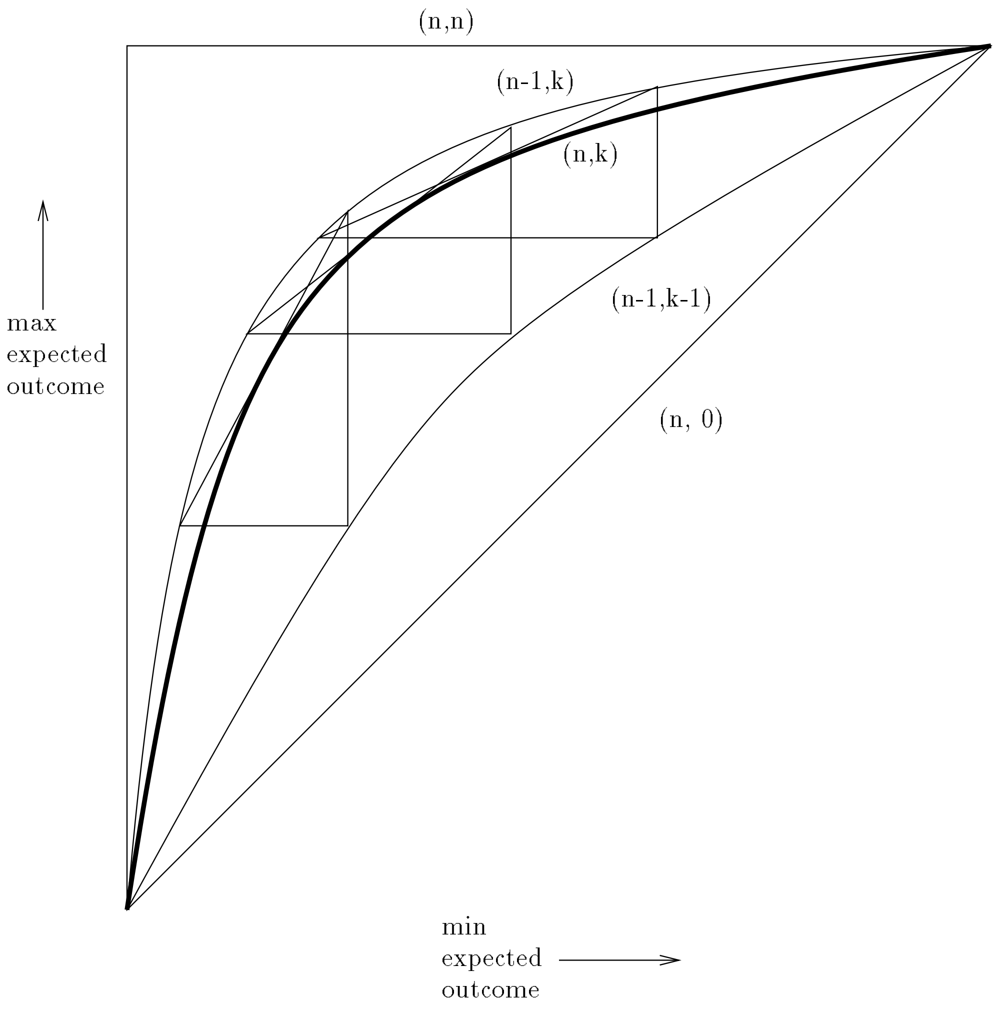

Figure 2: Effect of considering all choices of $G_\perp$. Each point on the $(n-1, k-1)$ curve corresponds to some possible default subgame $G_\perp$. The hypotenuse of the right triangle with corners on this point and the $(n-1, k)$ curve gives a set of games which dominate all other games with this fixed $G_\perp$. The set of optimal games with n voters and k failures is thus the minimum of the hypotenuses of all such right triangles.

Theorem 3 *Let G be a game of length n with outcome set $\{-1, +1\}$. Then for any $k \geq 0$, either $M_k G = 1$, $m_k G = -1$, or*

$$\tanh^{-1} M_k G - \tanh^{-1} m_k G \geq \frac{k}{2\sqrt{n}}. \tag{3}$$

With a little bit of algebra we can turn this into a lower bound on the length of a general coin-flipping game given lower bounds on the probabilities of any two distinct outcomes:

Corollary 4 *Let G be a coin-flipping game, let x be one of its outcomes, and let A range over k-bullet adversaries. If for some $\epsilon < \frac{1}{2}$, $\min_A \Pr[G \circ A = x] \geq \epsilon$ and $\min_A \Pr[G \circ A \neq x] \geq \epsilon$, then the maximum length of G is at least*

$$\frac{1}{16} \cdot \frac{k^2}{\ln^2\left(\frac{1}{\epsilon} - 1\right)}$$

Using a truncation argument, we can show that a similar result holds even if we are considering the *expected* length of G rather than its maximum length. The theorem below covers both the worst-case expected length (when the adversary is trying to maximize the running time of the protocol) and the best-case expected length (when the adversary is trying to minimize the running time of the protocol). The worst-case bound will be used later to get a lower bound on consensus. The proof of the theorem is given in the full paper.

Theorem 5 *Fix k, and let A range over k-bullet adversaries. Let G be a coin-flipping game with an outcome x such that $\min_A \Pr[G \circ A = x] \geq \epsilon$ and $\min_A \Pr[G \circ A \neq x] \geq \epsilon$. Then the worst-case expected length of G is at least*

$$\frac{3}{64} \cdot \frac{k^2}{\ln^2\left(\frac{1}{\epsilon/2} - 1\right)}$$

and the best-case expected length is at least

$$\frac{1}{32} \cdot \frac{\epsilon k^2}{\ln^2\left(\frac{1}{\epsilon/2} - 1\right)}.$$

For constant bias, Corollary 4 and Theorem 5 imply that we need $\Omega(t^2)$ local coin flips in both the worst and average cases. This is true even though the adversary's power is limited by the fact that (a) the local coin flips may have arbitrary ranges and distributions; (b) the adversary can hide coins, but cannot control them; (c) the adversary must decide which coins to hide or reveal immediately in an on-line fashion; and (d) the algorithm may observe and react to the choices of which coins to hide. These assumptions were chosen to minimize the power of the adversary while still capturing the essence of its powers in a distributed system with failures.

In contrast, it is not difficult to see that taking a majority of $\Theta(t^2)$ fair coins gives a constant bias even if (a) local coins are required to be fair random bits; (b) the adversary can replace up to t values with new values of its own choosing; (c) the adversary may observe the values of all the local coins before deciding which ones to alter; and (d) changes made by the adversary are invisible to the algorithm. So the $\Omega(t^2)$ lower bound for constant bias is tight for a wide range of assumptions about the powers of the algorithm and the adversary.[3]

2.3 Connection to Randomized Distributed Algorithms with Failures

The importance of coin-flipping games as defined above comes from the fact that they can often be found embedded inside randomized distributed algorithms. Let us discuss briefly how this embedding works.

Consider a randomized distributed algorithm in a model in which (a) all random events are internal to individual processes; and (b) all other nondeterminism is under the control of an adaptive adversary. Suppose further that the adversary has the power to kill up to k of the processes. Then given any randomized algorithm in which some event X that does not depend on the states of faulty processes occurs with minimum probability m and maximum probability M, we can extract a coin-flipping game from it as follows. Arbitrarily fix all the nondeterministic choices of the adversary except for the decision whether or not to kill each process immediately following each internal random event. (Since this step reduces the options of the adversary it can only increase m and decrease M.) Each step of the coin-flipping game corresponds to an execution of the distributed algorithm up to some such random event, which we interpret as the local coin. The adversary's choice to hide or reveal this local coin corresponds to its power to kill the process that executes the random event (thus preventing any other process from learning its value) or to let it run (which may or may not eventually reveal the value). The outcome of the coin-flipping game is determined by whether or not X occurs in the original system.

3 Lower Bound for Randomized Consensus

Consensus is a problem in which a group of n processes must agree on a bit. We will consider consensus in models in which at most t processes may fail by halting.

[3]The theorem does *not* apply if the adversary cannot observe local coin-flips, and so it cannot be used with an *oblivious* (as opposed to the usual *adaptive*) adversary. However, the bound on best-case expected length does imply that it is impossible to construct a "hybrid" constant-bias coin-flipping protocol that adapts to the strength of the adversary, finishing quickly against an oblivious adversary but using additional work to prevent an adaptive adversary from seizing control. This is not the case for consensus; for example, Chandra's consensus algorithm [Cha96] for a weak adversary switches over to an algorithm that is robust against an adaptive adversary if it does not finish in its usual time.

Processes that do not halt (i.e., *correct* processes) must execute infinitely many operations. (A more detailed description of the model is given in the full paper).

It is assumed that each process starts with some input bit and eventually *decides* on an output bit and then stops executing the algorithm. Formally, consensus is defined by three conditions:

- **Agreement.** All correct processes decide the same value with probability 1.
- **Non-triviality.** For each value v, there exists a set of inputs and an adversary that causes all correct processes to decide v with probability 1.
- **Termination.** All correct processes decide with probability 1.

Non-triviality is a rather weak condition, and for applications of consensus protocols a stronger condition is often more useful:

- **Validity.** If all processes have input v, all correct processes decide v with probability 1.

As non-triviality is implied by validity, if we show a lower bound on the total work of any protocol that satisfies agreement, non-triviality, and termination, we will have shown *a fortiori* a lower bound on any protocol that satisfies agreement, validity, and termination. Thus we will concentrate on consensus as defined by the first three conditions.

Since the agreement and termination conditions are violated only with probability zero, we can exclude all schedules in which they are violated without affecting the expected length of the protocol or the independence and unpredictability of local coin-flips. Thus without loss of generality we may assume that not only do agreement and termination apply to the protocol as a whole, but they also apply even if one conditions on starting with some particular finite execution α.

3.1 Overview of the Proof

In a randomized setting, we are concerned with the cost of carrying out a consensus protocol in terms of the expected total work when running against a worst-case adversary. We show how the coin-flipping lower bound can be used to show a lower bound on the worst-case expected cost of t-resilient randomized consensus in the standard asynchronous shared-memory model. As in the coin-flipping bound, we will measure the cost of a consensus protocol by the total number of local coin-flips executed by the processes. This measure is not affected by deterministic simulations, so any results we obtain for the shared-memory model will also apply to any model that can be simulated using shared memory, such as a t-resilient message-passing model.

For each adversary strategy and finite execution α there is a fixed probability that the protocol will decide 1 conditioned on the event that its execution starts with α. (We may speak without confusion of the protocol deciding 1, as opposed to individual processes deciding 1, because of the agreement condition.) For any set of adversaries, there is a range of probabilities running from the minimum to the maximum probability of deciding 1.

These ranges are used to define a probabilistic version of the bivalence and univalence conditions used in the well-known Fischer-Lynch-Paterson (FLP) impossibility proof for deterministic consensus [FLP85]. We will define an execution as *bivalent* if the adversary can force either outcome with high probability. A *v-valent* execution will be one after which only the outcome v can be forced with high probability. Finally, a *null-valent* execution will be one in which neither outcome can be forced with high probability. The notions of bivalence and v-valence (defined formally in the full paper) match the corresponding notions for deterministic algorithms used in the FLP proof; null-valence is new, as it cannot occur with a deterministic algorithm in which the probability of deciding each value v must always be exactly 0 or 1.

In outline, the proof that consensus is expensive for randomized algorithms retains much of the structure of the FLP proof. First, it is shown that with at least constant probability any protocol can be maneuvered from its initial state into either a bivalent or a null-valent execution. Once the protocol is in a bivalent execution, we show that there is a fair, failure-free extension that leads either to a local coin-flip or a null-valent execution. The result of flipping a local coin after a bivalent execution is, of course, random; but we can show that with high probability it leaves us with an execution which is either bivalent or null-valent or from which we are likely to return to a bivalent or a null-valent execution after additional coin-flips. If we do reach a null-valent execution, the coin-flipping bound applies.

Unlike a deterministic protocol, it is possible for a randomized protocol to "escape" through a local coin-flip into an execution in which it can finish the protocol quickly. But we will be able to show that the probability of escaping in this way is small, so that on average many local coin-flips will occur before it happens.

Details of the lower bound proof are given in the full paper. The result is:

Theorem 6 *Against a worst-case adaptive adversary, any t-resilient consensus protocol for the asynchronous shared-memory model performs an expected*

$$\Omega\left(\left(\frac{t-1}{\log(t-1)}\right)^2\right)$$

local coin-flips.

The bound counts the number of local coin-flips. Because we allow coin-flips to have arbitrary values (not just 0 or 1), local coin-flips performed by the same process without any intervening operations can be combined into a single coin-flip without increasing the adversary's influence. Thus the lower bound on local coin-flips immediately gives a lower bound on total work. Furthermore, because the coin-flip bound is not affected by changing the model to one that can be deterministically simulated by shared memory, we get the same lower bound on total work in any model that can be so simulated, no matter how powerful its primitives are. So, for example, wait-free consensus requires $\Omega(n^2/\log^2 n)$ work even in a model that supplies counters or $O(1)$-cost atomic snapshots.

4 Discussion

For those of us who like working with an adaptive adversary, randomization has given only a temporary reprieve from the consequences of Fischer, Lynch, and Paterson's impossibility proof for deterministic consensus with faulty processes. Theorem 6 means that even though we can solve consensus using randomization, we cannot hope to solve it quickly without a small upper bound on the number of failures, built-in synchronization primitives, or restrictions on the power of the adversary.

Fortunately, there are a number of natural restrictions on the adversary that allow fast consensus protocols without eliminating the faults that we might reasonably expect to observe in real systems. One plausible approach is to limit the knowledge the adversary has of register contents, to prevent it from discriminating against coin-flips it dislikes. Various versions of this can be found in the the consensus work of Chor, Israeli, and Li [CIL87] and Abrahamson [Abr88], and in the $O(n \log n)$ total work protocol of Aumann and Bender [AB96] and the $O(\log n)$ work-per-process protocol of Chandra [Cha96]. Restrictions on the amount of asynchrony can also have a large effect [AAT94, SSW91].

5 Acknowledgments

The author is indebted to Russell Impagliazzo for many fruitful discussions of coin-flipping problems, Steven Rudich for a suggestion that eventually became the truncation argument used to prove Theorem 5, Mike Saks for encouragement and pointers to related work, and Faith Fich, Wai-Kau Lo, Eric Ruppert, and Eric Schenk for many useful comments on an earlier version of this work.

References

[AAT94] Rajeev Alur, Hagit Attiya, and Gadi Taubenfeld. Time-adaptive algorithms for synchronization. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 800–809, Montréal, Québec, Canada, may 1994.

[AB96] Yonatan Aumann and Michael Bender. Efficient asynchronous consensus with a value-oblivious adversary scheduler. In *Proceedings of the 23rd International Conference on Automata, Languages, and Programming*, July 1996.

[Abr88] K. Abrahamson. On achieving consensus using a shared memory. In *Proceedings of the Seventh ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, August 1988.

[ADS89] Hagit Attiya, Danny Dolev, and Nir Shavit. Bounded polynomial randomized consensus. In *Proceedings of the Eighth ACM Symposium on Principles of Distributed Computing*, pages 281–294, August 1989.

[AH90] James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 11(3):441–461, September 1990.

[AN90] Noga Alon and Moni Naor. Coin-flipping games immune against linear-sized coalitions. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 46–54. IEEE, 1990.

[Asp93] James Aspnes. Time- and space-efficient randomized consensus. *Journal of Algorithms*, 14(3):414–431, May 1993.

[AW96] James Aspnes and Orli Waarts. Randomized consensus in $O(n \log^2 n)$ operations per processor. *SIAM Journal on Computing*, 25(5):1024–1044, October 1996.

[BOL85] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minimization of banzhaf values. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 408–416. IEEE, 1985.

[BOLS87] M. Ben-Or, N. Linial, and M. Saks. Collective coin flipping and other models of imperfect randomness. In *Combinatorics*, volume 52 of *Colloquia Mathematic Societatis János Bolyai*, pages 75–112, Eger (Hungary), 1987.

[BR90] Gabi Bracha and Ophir Rachman. Approximated counters and randomized consensus. Technical Report 662, Technion, 1990.

[BR91] Gabi Bracha and Ophir Rachman. Randomized consensus in expected $O(n^2 \log n)$ operations. In *Proceedings of the Fifth Workshop on Distributed Algorithms*, 1991.

[CFG+85] Benny Chor, Joel Friedman, Oded Goldreich, Johan Håstad, Steven Rudich, and Roman Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the 2th Annual Symposium on Foundations of Computer Science*, pages 396–407. IEEE, 1985.

[Cha96] Tushar Deepak Chandra. Polylog randomized wait-free consensus. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 166–175, May 1996.

[CI93] Richard Cleve and Russell Impagliazzo. Martingales with Boolean final value must make jumps of $O(1/n^{1/2})$ with constant probability. Unpublished manuscript, 1993.

[CIL87] B. Chor, A. Israeli, and M. Li. On processor coordination using asynchronous hardware. In *Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing*, pages 86–97, 1987.

[CL93] Jason Cooper and Nathan Linial. Fast perfect-information leader-election protocol with linear immunity. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 662–671. ACM, 1993.

[DDS87] D. Dolev, C. Dwork, and L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, January 1987.

[DHPW92] Cynthia Dwork, Maurice Herlihy, Serge Plotkin, and Orli Waarts. Time-lapse snapshots. In *Proceedings of Israel Symposium on the Theory of Computing and Systems*, 1992.

[FHS93] Faith Fich, Maurice Herlihy, and Nir Shavit. On the complexity of randomized synchronization. In *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing*, August 1993.

[FLP85] M. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed commit with one faulty process. *Journal of the ACM*, 32(2), April 1985.

[Fri92] Joel Friedman. On the bit extraction problem. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 314–319. IEEE, 1992.

[Har66] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1:385–394, 1966.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124–149, January 1991.

[LAA87] Michael C. Loui and Hosame H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. In Franco P. Preparata, editor, *Advances in Computing Research*, volume 4. JAI Press, 1987.

[LLS89] D. Lichtenstein, N. Linial, and M. Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9, 1989.

[Sak89] Michael Saks. A robust non-cryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics*, 2(2):240–244, 1989.

[SSW91] Michael Saks, Nir Shavit, and Heather Woll. Optimal time randomized consensus — making resilient algorithms fast in practice. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 351–362, 1991.

[Vaz85] Umesh Vazirani. Towards a strong communication complexity theory, or generating quasi-random sequences from two communicating slightly-random sources. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 366–378. ACM, 1985.