A Tight Lower Bound for Randomized Synchronous Consensus

Article · October 2002

DOI: 10.1145/277697.277733 · Source: CiteSeer

2 authors:

Ziv Bar-Joseph
Carnegie Mellon University
278 PUBLICATIONS 9,301 CITATIONS
SEE PROFILE

Michael Ben-Or
Hebrew University of Jerusalem
77 PUBLICATIONS 6,606 CITATIONS
SEE PROFILE

A Tight Lower Bound for Randomized Synchronous Consensus

Ziv Bar-Joseph[*] Michael Ben-Or[†]

April 8, 1999

Abstract

We prove tight upper and lower bounds of $\Theta(t/\sqrt{n \log(2+t/\sqrt{n})}$ on the expected number of rounds needed for randomized synchronous consensus protocols for a fail-stop, full information, adaptive adversary. In particular this proves that some restrictions are needed on the power of the adversary to allow randomized constant expected number of rounds protocols.

1 Introduction

Consensus is a fundamental problem in distributed computing in which a group of n processes must agree on a bit despite the interference of an adversary. In addition, trivial solutions (like always deciding on the same bit) are forbidden. The problem has been studied in various computation models: The synchronous message passing model, and the asynchronous (message passing and shared memory) models, and under various fault models or Adversaries: Fail-Stop and Byzantine, Static and Adaptive, Computationally Bounded and Unbounded Adversaries - just to name a few.

In the synchronous setting it is well known that if the adversary can halt up to t processes, then any deterministic agreement protocol may need at least $t+1$ communication rounds to reach agreement, (see for example [Lyn96]) and efficient $t+1$ round agreement protocols are known even for Byzantine adversaries [GM93].

Assuming reasonable bounds on the power of the adversary there are synchronous randomized agreement protocols that require only constant expected number number of rounds to reach agreement [CMS89, Rab83, FM97]. Without such limitations much less is known. In fact for the strong adversary (i.e. computationally unbounded adaptive Byzantine adversary) that has full information about the system (including full knowledge of the local variables of all the processes) fast $O(1)$ randomized solution exist only for $t = O(\sqrt{n})$ [BO83], but for larger t the best known randomized solution is the deterministic $t+1$-round protocol! Furthermore, no lower bounds are known for randomized synchronous agreement protocols. It is therefore unclear what limitations on the power of the adversary are necessary to guarantee $O(1)$ randomized solutions.

1.1 Our Results

The main result of this paper is a lower bound on the number of rounds needed to reach agreement by any synchronous randomized agreement protocol, that acts in the presence of an adaptive, full-information and computationally unbounded, fail-stop adversary. If the adversary can fail up to t of the n processes, we show that with high probability the adversary can force the protocol to make $\Omega(t/\sqrt{n \log n})$ rounds. For $t = \Omega(n)$ this yields a lower bound of $\Omega(\sqrt{n/\log n})$ on the expected number of rounds to reach agreement.

[*]This work was done as part of a B.Sc. thesis under the supervision of Michael Ben-Or.

[†]Research partially supported by a U.S. Air Force grant, by an Israel Science Foundation grant, and by the Minerva Leibniz Center for Research in Computer Science

In particular this lower bound shows that some limitations on the power of the adversary are needed to allow $O(1)$ round randomized agreement protocols.
A simple variation of the randomized agreement protocol in [BO83] provides an $O(t/\sqrt{n})$ expected number of rounds protocol for our fail-stop adversary leaving an annoying $\sqrt{\log n}$ gap between the upper and lower bound. Such a gap seems natural because our lower bound argument works round by round. In the upper bound protocol the expected number of processes that the adversary has to halt is only $\sqrt{n}$ per round, but for our lower bound argument we need to halt $\sqrt{n \log n}$ to guarantee an error probability smaller than $n^{-1/2}$ at each round.

Our lower bound argument relies on the fact that an adaptive fail-stop adversary can easily bias one round collective coin flipping games. These games are defined and studied in the next section where we show that the adversary can always bias the coin towards 0 or towards 1 (but not necessarily both) by halting just a few processes. This is best possible since simple fail-stop collective coin flipping games such as 0-1 majority voting, where any missing value is counted as 0, can be biased only towards 0 and not towards 1 by a fail-stop adversary.

Incorporating such "one-side-bias" coin flipping to Ben-Or's [BO83] protocol we obtain an agreement protocol that takes an expected $O(\sqrt{n/\log n})$ rounds in the presence of any adversary that can fail-stop any number of processes, proving that our lower bound is tight.
Randomized agreement protocols have been presented under the assumption that $t < n/2$. In the synchronous fail-stop environment agreement is possible for any $t < n$. The agreement protocols SynRan presented in Section 4, works correctly for any t, $0 \leq t \leq n$ and the expected time to reach agreement is $O(t/\sqrt{n \log(2 + t/\sqrt{n})})$.

1.2 Related Work

We shall not attempt to review here all the known randomized distributed agreement protocols, and we direct the reader to the survey paper by Chor and Dwork [CD89].

Aspnes [Asp97] recently proved an $\Omega(t^2/\log^2 t)$ lower bound on the number of coin flips for the asynchronous consensus problem when the adversary can fail up to t processes. His argument extends and follows the lines of the proof of the well known result that there are no fault-tolerant deterministic asynchronous agreement protocols [FLP85]. We note that Aspnes' results cannot be applied to the synchronous setting, and as mentioned above, no previous lower bounds on the expected number of rounds were known for the synchronous model. Nonetheless our work borrows several important ideas from Aspnes' paper.

Several results are known for non-adaptive full-information adversaries. Chor, Merritt and Shmoys [CMS89] provide a randomized $O(1)$ expected number of rounds protocol for non-adaptive fail-stop adversaries. In particular this shows that our lower bound does not hold without the adaptive selection of the faulty processes.

Proving tight bounds on the expected number of rounds for full-information non adaptive Byzantine adversaries is an important open problem. The best upper bound known here is the $O(t/\log n)$ expected number of rounds protocol of Chor and Coan [CC85], but from what we currently know one cannot rule out the possibility that $O(1)$ expected number of rounds protocols exist even for $t = \Omega(n)$.

Many varieties of collective coin-flipping games have been studied, (see [BOL89, Lin94] for a survey). Aspnes (in [Asp97]) was the first to study multi-round coin flipping games in the fail-stop adversary model. From his results one can conclude that by halting $O(\sqrt{n} \log n)$ processes the adversary can bias the game to one of the possible outcomes with probability greater than $(1 - 1/n)$. In the next section we prove a slightly sharper result for the much simpler case of one round coin flipping games needed here.

2 One Round Coin Flipping Game

A collective coin flipping game is an algorithm for combining many local random variables to obtain a global coin [BOL89]. We consider here only one round coin flipping games with an adaptive fail-stop adversary. The most general game G of this sort can be described as follows. Each of the n participants in the game selects a single input variable. The input variables are drawn from independent probability distributions whose range and distribution can be arbitrary. After seeing all the input variables, an adaptive t-adversary can hide up to t of the input variables by replacing their value with a special default value $-$. At this point we obtain a sequence of values, some of which are the original input values, and at most t of which are $-$. We apply a function

$$f : \{X_1 \cup -\} \times \cdots \times \{X_n \cup -\} \to \{1..k\}$$

(where X_i is the range of the i-th local variable) to this sequence and obtain the outcome of the game. We say that the function f determines the game G.

2.1 Control Over One Round Games

We say that a t-adversary controls a one round coin flipping game if the adversary's strategy can guarantee a particular outcome with probability greater than $1 - 1/n$. As mentioned above, it is easy to construct non trivial games in which the adversary cannot bias the results in both directions (extending the 0-1 majority function to consider $-$ as 0, for example). It is easy to show however that for $t = k4\sqrt{n \log n}$, a t-adversary can always control the game.

Let $X^n = X_1 \times \cdots \times X_n$ be be the product probability space of the X_i. For $y = (y_1, \ldots, y_n) \in X^n$, $s \subseteq \{1 \ldots n\}$ define $y_{\overline{s}}$ to be the sequence obtained by replacing the value y_i by the default value $-$ at all coordinates $i \in s$ of y.

For each $v \in \{1..k\}$ let

$$U^v = \{y \in X^n \mid \forall s, s \subseteq \{1 \ldots n\}, |s| \leq t \longrightarrow f(y_{\overline{s}}) \neq v\},$$

that is, U^v is the set of points in X^n for which a t-adversary cannot change the outcome of the game to v. Define $h = 4\sqrt{n \log n}$.

Finally, for $A \subseteq X^n$ and $l \geq 0$ we define $B(A, l)$ to be the collection of all elements in X^n differing from some element in A by at most l coordinates.

Lemma 2.1 Let f be a function defining a one round coin flipping game on a distribution space X^n with k ($k < \sqrt{n}$) possible outcomes and let $t \geq k4\sqrt{n \log n}$. Then there exists a $v \in \{1..k\}$ so that $\Pr(U^v) < 1/n$.

Proof: Assume by contradiction that there is no such v. Schechtman's theorem [Sch81] states that for $A \subseteq X^n$ with $\Pr(A) = \alpha$ and $l \geq l_0 = 2\sqrt{n \log(\alpha^{-1})}$ we have $\Pr(B(A, l)) \geq 1 - e^{(-(l-l_0)^2/4n)}$.
Here we have $\Pr(U^v) \geq 1/n$ for every v, so from the theorem we get
$\Pr(B(U^v, h)) = \Pr(B(U^v, 4\sqrt{n \log n})) \geq 1 - e^{(-(4\sqrt{n \log n} - 2\sqrt{n \log n})^2/4n)} = 1 - e^{-\log n} = 1 - 1/n$.
So if we define $B^v = B(U^v, h)$ we get for every v, $\Pr(B^v) \geq 1 - 1/n$. Since $k < \sqrt{n}$ it means that $\cap_v B^v \neq \emptyset$, so there exists $y \in \cap_v B^v$. Since $y \in \cap_v B^v$ there exists $x^1 \in U^1$ that differs from y in at most h coordinates. Define s_1 to be the set of coordinates in which x^1 and y differ. Then $y_{\overline{s_1}} = x^1_{\overline{s_1}}$. Following this line there is $x^2 \in U^2$ that differs from y in at most h coordinates, so $y_{\overline{s_2}} = x^2_{\overline{s_2}} \Rightarrow y_{\overline{s_1 s_2}} = x^1_{\overline{s_1 s_2}} = x^2_{\overline{s_1 s_2}}$. We can continue this and in the end we will get $k+1$ variables $y_{\overline{s_1..s_k}} = x^1_{\overline{s_1..s_k}} = .. = x^k_{\overline{s_1..s_k}}$. Note however that since $x^1_{\overline{s_1..s_k}}$ differs from x^1 in at most $kh = k4\sqrt{n \log n} \leq t$ coordinates, it means that $f(x^1_{\overline{s_1..s_k}}) \neq 1$ and for that same reason, $f(x^1_{\overline{s_1..s_k}}) = f(y_{\overline{s_1..s_k}}) \notin \{2..k\}$. This is a contradiction since in $y_{\overline{s_1..s_k}}$ there are at most t $-$ so f should be defined on such variable. So we can conclude that there must be a v such that $\Pr(U^v) < 1/n$. •

Corollary 2.2 In an n player one round coin flipping game with k possible outcomes, if the adversary can fail-stop more than $k4\sqrt{n \log n}$ players, he can bias the result towards one particular outcome with probability greater than $1 - 1/n$.

Proof: From the previous lemma there exists $v \in \{1..k\}$ so that $\Pr(U^v) < 1/n$. If $y \notin U^v$ then there exists a subset $s \subset \{1 \ldots n\}, |s| \le k4\sqrt{n \log n}$ so that $f(y_{\bar{s}}) = v$, so by failing the players in s the adversary can bias the result to v. Since $\Pr(y \notin U^v) \ge 1 - 1/n$ the corollary is proved. •

3 The Consensus Lower Bound

3.1 The Model

A synchronous distributed system is a collection of processes that communicate with each other by sending messages between them, where the computations proceeds in synchronous rounds of local computation (including local random coin flips) and message exchange.
Without loss of generality we can divide each round of the computation to two synchronous phases

Phase A – Generating the local coins (if needed) and local computation.

Phase B – Sending and receiving messages.

We model faulty behavior of the processes by an Adversary. The adversary we consider here, is the fail-stop, adaptive-strongly-dynamic, computationally-unbounded adversary (as described in the survey [CD89]). At the beginning of each round all the active processes execute "Phase A" of the round and the adversary can examine their local coins and variables, and the messages they wish to send. Based on this information the adversary can fail-stop some processes during the message exchange phase of the round. If a process fails during the message exchange phase the adversary can decide which subset of its messages will be sent. (We could also assume that messages are sent out according to some order and if the adversary fails a message of some process all later messages of that process will not be sent.) Moreover a process that has not sent all its messages in round k, will not sent any more messages in future rounds and is considered to be "dead" or faulty. A t-adversary is an adversary that can fail at most t processes. Apart from process failing, we assume that the communication links are perfectly reliable – all the messages that are sent are delivered.
In this paper we study randomized solutions to the Consensus problem in a synchronous distributed system with a fail-stop adversary.

Consensus is a distributed computation problem in which a group of processes, $P_1, \ldots, P_n$, where each P_i has an input bit $x_i \in \{0, 1\}$, have to agree on a common output bit despite the intervention of an adversary. We assume that during the computation each process decides on its output value and cannot change its decision.

A distributed algorithm is a t-resilient solution to the consensus problem if for any adversary that can fail at most t processes the algorithm satisfies the following three conditions:

- Agreement. All non faulty processes decide the same value with probability 1.
- Validity. If all processes have the same initial value v, then v is the only possible decision value.
- Termination. All non faulty processes decide with probability 1.

We measure the complexity of an execution of a consensus algorithm by the number of rounds taken until all the non faulty processes decide. The complexity of a t-resilient consensus algorithm is the maximal expected number of rounds to reach agreement when running the algorithm under the control of all t-adversaries. The main result of this section is a lower bound on the complexity of t-resilient consensus algorithms. Our proof below shows that it is enough to fail at most $O(\sqrt{n \log n})$ processes in each round to be sure that with high probability the algorithm will continue to the next round so there is a t-adversary that can force the algorithm with high probability to make $\Omega(t/\sqrt{n \log n})$ rounds.

3.2 Definitions

We will mark the round we are in k, α_k will denote the state of the execution at the beginning of round k, and t' will denote the number of the processes the adversary can still fail.
For each execution α_k and adversary b, let $\Pr[v \mid \alpha_k, b]$ denote the probability in state α_k that the protocol decides v running under the control of b. For any set of adversaries C let $r_{v,C}(\alpha_k)$ be the set of probabilities ranging over all adversaries in C; that is, $r_{v,C} = \{\Pr[v \mid \alpha_k, c] \mid c \in C\}$. From the fact that the protocol terminates with probability 1, we know that that $\Pr[0 \mid \alpha_k, c] = 1 - \Pr[1 \mid \alpha_k, c]$. So by keeping track of $r_{1,c}$ alone we do not lose any information.
For our lower bound proof we are interested only in adversaries that fail no more than $4\sqrt{n \log n}\ + 1$ processes at each round. We denote this group of adversaries by B, and from now on we will limit our discussion to adversaries in B. To simplify the noation we use $r(\alpha_k)$ to denote $r_{1,B}(\alpha_k)$.
Following the notation in [Asp97] we define a probabilistic version of univalence, bivalence and null-valence. We will classify executions using the maximum and minimum probabilities of deciding 1 according to the following table.

Classification of α_k	$\min r(\alpha_k)$	$\max r(\alpha_k)$
bivalent	$< 1/\sqrt{n} - k/n$	$> 1 - 1/\sqrt{n} + k/n$
0-valent	$< 1/\sqrt{n} - k/n$	$\leq 1 - 1/\sqrt{n} + k/n$
1-valent	$\geq 1/\sqrt{n} - k/n$	$> 1 - 1/\sqrt{n} + k/n$
null-valent	$\geq 1/\sqrt{n} - k/n$	$\leq 1 - 1/\sqrt{n} + k/n$

Note that this classification is exhaustive: every execution falls into exactly one of these classes. An execution that is either 0-valent or 1-valent will be called univalent.
The use of this classification as follows. We will show that if an execution enters a null-valent or bi-valent state then, with high probability, the adversary can force the execution to remain in a null-valent or bi-valent state by failing at most $4\sqrt{n \log n}\ + 1$ processes per round. We shall also prove that the adversary can always select the initial state of the algorithm to be null or bi-valent. Thus with high probability the adversary can keep the computation going for at least $t/(4\sqrt{n \log n}\ + 1)$ rounds before reaching the limit on the number of faulty player.
We now describe adversary's strategy in each one of those states, and by implementing this strategy we will get the lower bound.

3.3 Null-Valent Execution

Lemma 3.1 An execution α_k that is null-valent, will stay null-valent at the end of round k with probability higher than $1 - 1/n$ by killing at most $4\sqrt{n \log n}$ processes.

Proof: As described, at phase A of round k we have a local coin flip at each process (if needed). This means that at most n coins are flipped at the beginning of this round. Then we have the message passing. Divide the results of the coin flips at the beginning of round k into three different results:

1. After the coin flips the execution becomes bivalent or 1-valent.

2. After the coin flips the execution becomes 0-valent.

3. After the coin flips the execution remains null-valent.

Since those three possibilities are exhaustive, according to lemma 2.1 we can bias the result of the coin flipping game to at least one of this three outcomes (by taking over at most $4\sqrt{n \log n}$ players), with probability higher than $1 - 1/n$. Assume we can bias the result to (1). If the execution becomes bivalent

or 1-valent, It means that by killing at most $4\sqrt{n\log n}$ processes at round k we will be at the end of round k in an execution α_{k+1} such that
$\max r(\alpha_{k+1}) \geq 1 - 1/\sqrt{n} + (k+1)/n$ with probability higher than $1 - 1/n$. So at the beginning of round k we have $\max r(\alpha_k) \geq (1-1/n)(1 - 1/\sqrt{n} + (k+1)/n) \Rightarrow \max r(\alpha_k) > 1 - 1/\sqrt{n} + k/n$ meaning α_k is either 1-valent or bivalent. This contradicts the fact that α_k is a null-valent execution. On the other hand, the same proof (only this time by using $\min r(\alpha_k)$) shows that if we can bias the result to (2) then α_k is either 0-valent or bivalent contradicting the fact that it is null-valent. Combining this two observations, we get that the only possible outcome that we can force is (3), so with probability higher than $1-1/n$ the execution remains null-valent.

●

Corollary 3.2 An execution α_k which is null-valent, will take at least $\frac{t(k)}{12\sqrt{n\log n}}$more rounds with probability higher than $(1 - \frac{1}{\sqrt{n\log n}})$.

Proof: According to the previous lemma, if we are in a null-valent execution, the execution will stay null-valent with high probability. But if we are in a null-valent execution the algorithm does not end (since when the algorithm ends it is in a univalent execution). In order to use the previous lemma we must fail at most $4\sqrt{n\log n}$ processes each round, so the probability that the execution ended before the end of round $\frac{t(k)}{12\sqrt{n\log n}}$ is $(1-1/n)^{\frac{t(k)}{12\sqrt{n\log n}}}$ and since $t(k) \leq n$ the corollary is proved. ●

3.4 Bivalent Execution

We will describe the strategy that the adversary should implement in order to continue the algorithm if the current sate α_k is bivalent. First, the adversary allows all processes to flip coins (if needed). Then we check the resulting execution if all the messages in round k would have been sent. If by sending all messages the execution becomes bivalent or null-valent we pass all the messages and continue to the next round. Otherwise, by passing all the messages the execution becomes univalent (assume 1-valent). Now, since we started with a bivalent execution, there is a strategy that by failing up to $4\sqrt{n\log n}+1$ processes at each round leads us to $\min r(\alpha_k) \leq 1/\sqrt{n}$. The adversary will implement this strategy step by step and inspect the state of the execution after each step. At each step in this process the adversary may fail a process but send all its messages, then step by step fail the messages this process should not send according to our minimizing strategy. We continue this process until one of the following happens:

1. If at some point we move to a bivalent or null-valent state α_{k+1} then stop failing the rest to stay in either of these states.

2. If by failing the next process (while sending its messages) the execution turns from 1-valent to 0-valent, we shall not fail this process and send all its messages. Note that failing the sender or not at the beginning of the next round switches the state from 1-valent to 0-valent so the state itself is bivalent.

3. If by failing the next message the execution turns from 1-valent to 0-valent then fail this message and stop failing the rest. We are in a 0-valent state. We claim that failing the receiver at the beginning of round $k+1$ the state cannot remain 0-valent. This is so because failing the receiver we remain in the same state regardless of whether the last message was sent or not. By our assumption by sending the message we are in a 1-valent state say α_{k+1}, with $\min r(\alpha_{k+1}) \geq \epsilon_{k+1}$ (for the appropriate ϵ_{k+1}). Failing the receiver is included among the adversary strategies defining this minimum so failing the receiver we are definitely not in a 0-valent state. If by failing the receiver we move to a null valent

state then we fail the receiver to remain in this state. Otherwise failing the receiver or not switches the state from 1-valent to 0-valent so we are in a bivalent state.

4. Finally, if by fully implementing this strategy the execution remains 1-valent, the execution becomes univalent and the adversary will act in the following way.

3.5 Entering a Univalent State

As long as our adversary has not run out of processes to fail, the adversary will arrive at a univalent state, say 1-valent, at the beginning of a round only while implementing the strategy for $\min r(\alpha_k)$ for some previous round k. As long as the state remains 1-valent we will continue to implement this strategy in the way described at the previous section (failing one process or message at a time and examining the effect it has on the execution) until one of the following happens :

1. As described in the previous section, failing some of the messages and sending others will turn the execution to null-valent or bivalent. If this happens, we will continue with the strategy for this kind of executions as described in one of the previous sections.
2. The execution becomes 0-valent after phase A of round m ($m > k$) because of coin flips. If this happens, since we were in a 1-valent execution at the beginning of this round, we will start implementing the strategy for $\max r(\alpha_m)$. If the execution remains 0-valent at the end of this round, we shall treat it as a univalent (but this time 0-valent) as described in this section.
3. The algorithm ends with 1.

Lemma 3.3 If an execution turns to be univalent (assume 1-valent), the probability that it will stay 1-valent and end with 1 before the adversary runs out of processes to fail is less than $1/\sqrt{n}$.

Proof: Since at the beginning of round k the execution was bivalent or 0-valent meaning $\min r(\alpha_k) \leq 1/\sqrt{n}$, and from that round on we used the strategy for $\min r(\alpha_k)$, it means that the probability to finish early with 1 is less than $1/\sqrt{n}$. •

Proving the lemma leads us to a more general corollary dealing with bivalent executions.

Corollary 3.4 Let α_k be a bivalent execution and let t' be the number of remaining process that the adversary can fail, then under our adversary the execution will continue for at least $t'/(4\sqrt{n \log n} + 1)$ more rounds with probability greater than $(1 - 1/\sqrt{\log n})$.

Proof: If the execution becomes null-valent then the lemma is proved from corollary 3.2. Otherwise, an execution can terminate only by passing through a univalent state (since in a bivalent state the probabilities of 0 and 1 are still positive). So, in order to end the execution must become v-univalent at some round $m \geq k$ for some $v \in 0, 1$. By the previous lemma the probability to end early in this state is less than $1/\sqrt{n}$. Assuming that this does not happen the execution can return to a bivalent, null-valent, $(1 - v)$-univalent at later round. Thus for each future round the probability that the adversary will fail to continue the execution is bounded by $1/\sqrt{n}$.

Implementing the strategy for a bivalent execution we fail at most $4\sqrt{n \log n} + 1$ processes each round, the probability of ending the execution in less than $t'/(4\sqrt{n \log n} + 1)$ rounds is less than $(1/\sqrt{n})(t'/(4\sqrt{n \log n} + 1))$ proving the corollary since $t' \leq n$. •

In order to use this corollary for the lower bound, we must show that there is an initial state that is either univalent or bivalent, and this is done in the next section.

3.6 Initial State

Lemma 3.5 Any synchronous consensus algorithm has an initial state which by failing at most one process becomes null-valent or bivalent.

Proof: If there is an initial state which is null-valent or bivalent we are done. Otherwise, in every initial state the algorithm is in a univalent state. Consider two initial states α_1 and β_1 such that $\min r(\alpha_1) = 0$ and $\max r(\beta_1) = 1$ (such states must exist due to the validity condition). There exists a chain of initial states $\alpha_1 = \alpha_1^1, \ldots, \alpha_1^k = \beta_1$ such that adjacent pair of states differ in only one input. Therefore we have an i such that α_1^i is 0-valent and α_1^{i+1} is 1-valent and the difference between these two initial states is the input of process P. Now check the state of execution α_1^i if we fail process P at the beginning of round 1. If the execution becomes null-valent or bivalent we are done. Otherwise, the execution remains univalent. If it stays 0-valent then if we start in initial state α_1^{i+1} we get a bivalent execution by failing one more process, and if it becomes 1-valent then in the initial state α_1^k the execution is bivalent, by failing one more process. Both ways the lemma is proved. •

After proving that we can always find an initial state that fits our strategy, we finally have

Theorem 1 For any t-resilient randomized synchronous consensus algorithm there is an adaptive fail-stop stop t-adversary that can force the algorithm to run for at least $\Omega(t/(\sqrt{n}\log n))$ rounds with probability greater than $1 - 1/\sqrt{\log n}$.

Corollary 3.6 If the adversary can fail $t = \Omega(n)$ processes, then the adversary can force any t-resilient randomized synchronous consensus algorithm to perform at least $\Omega(\sqrt{n/\log n})$ rounds before reaching agreement with probability greater than $1 - 1/\sqrt{\log n}$.

4 Consensus Algorithm

To prove that the lower bound presented in the previous section is tight, we present a new randomized synchronous consensus algorithm that for any t, $t = \Omega(n)$ reaches agreement in $O(t/\sqrt{n \log n})$ expected number of rounds under any fail-stop t-adversary. The algorithm is similar to Ben-Or's algorithm [BO83], but to raise the immunity to fail-stop failures we use "one-side-bias" coin flipping function instead of the symmetric coin flipping used in the original algorithm.
Let $x_i \in \{0, 1\}$ be the input of process P_i. We denote by b_i the process' current choice for the consensus value (initialized to x_i). In particular when the process ends the algorithm, b_i contains its decision value. The algorithm proceeds in rounds, and the index of the current round is stored in r. We denote by O_i^r (Z_i^r) the number of 1's (0's) received by process P_i at round r (including b_i). We denote by N_i^r the number of messages that process P_i received in round r.

```
SynRan{ (* algorithm for process $P_i$ *)
    $r$ :=1
    $N_{-1}^r = N_0^r = n$
    $b_i = x_i$
    decided = FALSE
    WHILE TRUE DO {
        send $b_i$ to all processes
        receive all messages sent to $P_i$ in round $r$
        compute $O_i^r$, $Z_i^r$, $N_i^r$
        If($N_i^r < \sqrt{n/\log n}$) {
            send $b_i$ to all processes
            receive all messages sent to $P_i$ in round $r+1$
            implement the deterministic protocol for $\sqrt{n/\log n}$ rounds
        }
        If(decided = TRUE) {
            diff = $N_i^{r-3} - N_i^r$
            IF (diff $\leq N_i^{r-2}/10$) {
                STOP
            } ELSE {
                decided = FALSE
            }
        }
        IF $O_i^r > (7N_i^{r-1})/10$ THEN $b_i = 1$, decided = TRUE
        ELSE IF $O_i^r > (6N_i^{r-1})/10$ THEN $b_i = 1$
        ELSE IF $Z_i^r = 0$ THEN $b_i = 1$
        ELSE IF $O_i^r < (4N_i^{r-1})/10$ THEN $b_i = 0$, decided = TRUE
        ELSE IF $O_i^r < (5N_i^{r-1})/10$ THEN $b_i = 0$
        ELSE set $b_i$ to 0 or 1 with equal probability
        $r := r + 1$
    }
}
```

4.1 Correctness

To prove the correctness of the algorithm we use the following notations:
We say that P_i decides v in round r if P_i sets b_i to v in round r and sets decided to TRUE. We say that

P_i proposes v in round r, if P_i sets b_i to v in round r without flipping a coin.
We denote the rounds in which the number of living processes is greater than $\sqrt{n/\log n}$ the probabilistic stage of the algorithm. The remaining rounds are called the deterministic stage.
Correctness follows from the following three lemmata.

Lemma 4.1 If at the end of round r of the probabilistic stage all active processes have the same value v for b_i, then all active processes will eventually decide on v.

Proof: Since all the messages sent in the following round are the same (v), then according to the algorithm, the only possible value that each process can propose is v. So v is the only possible decision for him for the following round. This situation continues in all the following rounds (since the adversary can only fail processes and cannot add new messages). If the adversary doesn't fail any more processes the algorithm will end in the next round. If the adversary doesn't want this to happen it must fail 1/10 of the remaining processes every 4 rounds, and eventually all living processes will start implementing the deterministic protocol and the algorithm will end deciding on v (the only possible value). •

Lemma 4.2 If a process stops at round $r+1$ after it decided v at round r of the probabilistic stage, all other processes will eventually decide v.

Proof: Let P_i be the process that stopped at round $r+1$ after it decided at round r, and assume $v=1$.
claim: At most $N_i^{r-1}/10$ process have failed in round $r-1$ and round r
proof: Denote by N_{start}^k the number of processes at the beginning of round k, and by N_{end}^k the number of processes at the end of round k. We get $N_{start}^{r-1} \leq N_i^{r-2}$ since all processes that P_i didn't get there messages in round $r-2$ must have failed in that round or in a previous one. On the other hand $N_{end}^r \geq N_i^{r+1}$ since all processes that P_i did get there messages in round $r+1$ must have been alive at round r. So $N_{start}^{r-1} - N_{end}^r \leq N_i^{r-2} - N_i^{r+1} \leq N_i^{r-1}/10$ since P_i stopped at round $r+1$, proving the claim.
Now for each process P_j, if $N_j^{r-1} \leq N_i^{r-1}$ then since P_i decided 1 at round r, it means he received more than $7N_i^{r-1}/10$ messages of 1 at that round. Since at most at most $N_i^{r-1}/10$ process has failed in round r, it means that P_j received more than $6N_i^{r-1}/10$ messages of 1 in round r. Since $N_j^{r-1} \leq N_i^{r-1}$, P_j received more than $6N_j^{r-1}/10$ messages of 1 in round r and so P_j proposes 1 for the next round.
If $N_j^{r-1} > N_i^{r-1}$ denote by $k = N_j^{r-1} - N_i^{r-1}$ the difference. From the claim we know that at most $N_i^{r-1}/10$ processes have failed in round $r-1$ and round r. In addition, the k processes that P_j sees at round $r-1$ and P_i doesn't see in that round have surely failed in round $r-1$, so at most $N_i^{r-1}/10 - k$ processes have failed in round r. Since P_i decided 1 at round r, it means he received more than $7N_i^{r-1}/10$ messages of 1 at that round. Combining this two observations we can conclude that all the processes received at least
$7N_i^{r-1}/10 - (N_i^{r-1}/10 - k) = 6N_i^{r-1}/10 + k$
messages of 1 in round r. Since $6N_i^{r-1}/10 + k > 6(N_i^{r-1} + k)/10 = 6N_j^{r-1}/10$,
P_j proposes 1 for the next round. So in both cases P_j proposes 1 for the next round. This is true for every processor so according to the previous lemma all processes will eventually decide 1. The case where $v=0$ is symmetric, so the lemma is proved. •

Following [GP90] we also concatenate the probabilistic stage with a deterministic stage. Unlike their work, in our algorithm the beginning of the deterministic stage doesn't depend on the round number (which is a number that all processes share in common), but rather on the number of living processes. Since not all processes see the same number of messages each round, we need the following lemma to prove that this concatenation doesn't violate the correctness condition.

Lemma 4.3 If a process j implements the deterministic protocol from round $r+1$, then all processes that didn't fail will have the same value at the end.

Proof: If j starts implementing the deterministic protocol at round $r+1$ it means that $N_j^r < \sqrt{n/\log n}$ (note the one round delay between stopping the probabilistic protocol and starting the deterministic one). So for every process i $N_i^{r+1} < \sqrt{n/\log n}$. If there is a process i that stopped at round $k \leq r$ deciding v, it means (according to the previous lemma) that all active processes have proposed v for round k, and so v is the only possible decision value for the deterministic protocol. Otherwise, no process stopped before round $r+1$, and since $N_i^{r+1} < \sqrt{n/\log n}$ for every process i, all processes will start implementing the deterministic protocol. No process can stop at round $r+1$, since for every process i we get $N_i^{r+1} \leq N_j^r < \sqrt{n/\log n}$. Since the algorithm checks the number of received messages (in order to decide if to start implementing the deterministic protocol) before it checks if it must stop in this round, it means that all processes will start implementing the deterministic protocol from this round. Note that once a process starts implementing the deterministic protocol, then for each i, b_i remains the same from now on (due to the one round delay). This ensures us that all processes will end up deciding the same value, proving the lemma. •

4.2 Bounding the Number of Rounds

To bound the complexity of the algorithm, we need the following large deviation lemma. We present the proof since we could not find a handy non asymptotic reference.

Lemma 4.4 Let x be the number of 1-s in an n player coin flipping game, where each player flips an independent unbiased 0-1 coin. Then for $t < \sqrt{n}/8$

$$\Pr(x - E(x) \geq t\sqrt{n}) \geq \frac{e^{-4(t+1)^2}}{\sqrt{2\pi}}.$$

Proof: From the Sterling approximation we know that $\binom{n}{\frac{n}{2}} \geq \frac{2^n}{\sqrt{2\pi n}}$.
Now $\frac{\binom{n}{\frac{n}{2}+l+1}}{\binom{n}{\frac{n}{2}+l}} = \frac{n-2l}{n+2l+2} = 1 - \frac{4l+2}{n+2l+2} \geq 1 - \frac{4l+2}{n} \Longrightarrow \binom{n}{\frac{n}{2}+s} \geq \binom{n}{\frac{n}{2}} \prod_{l=0}^{s-1}(1 - \frac{4l+2}{n})$.

Using the fact that for $0 < x < 1/2$ we have $(1-x) \geq e^{-2x}$, we get for $s < n/8$:
$\binom{n}{\frac{n}{2}} \prod_{l=0}^{s-1}(1 - \frac{4l+2}{n}) \geq \binom{n}{\frac{n}{2}} e^{\frac{-2}{n}\sum_{l=0}^{s-1}(4l+2)} = \binom{n}{\frac{n}{2}} e^{\frac{-4s^2}{n}} \Longrightarrow \binom{n}{\frac{n}{2}+t\sqrt{n}} \geq \binom{n}{\frac{n}{2}} e^{-4t^2}$.
Now we can estimate the probability that $(x - E(x) \geq t\sqrt{n})$.
$\Pr(x - E(x) \geq t\sqrt{n}) \geq 2^{-n} \sum_{i=t\sqrt{n}}^{\frac{n}{2}} \binom{n}{\frac{n}{2}+i}$. Using our estimation we get
$\sum_{i=t\sqrt{n}}^{\frac{n}{2}} \binom{n}{\frac{n}{2}+i} \geq \sum_{i=t\sqrt{n}}^{(t+1)\sqrt{n}} \binom{n}{\frac{n}{2}+i} \geq \sqrt{n}\binom{n}{\frac{n}{2}+(t+1)\sqrt{n}} \geq \binom{n}{\frac{n}{2}} \sqrt{n} e^{-4(t+1)^2} \geq \frac{2^n}{\sqrt{2\pi n}} \sqrt{n} e^{-4(t+1)^2}$ by the Sterling approximation stated above. Therefore
$\Pr(x - E(x) \geq t\sqrt{n}) \geq \frac{\sqrt{n}e^{-4(t+1)^2}}{\sqrt{2\pi n}} = \frac{e^{-4(t+1)^2}}{\sqrt{2\pi}}$ proving the lemma. •

Applying this lemma for $t = \sqrt{\log n}/8$ we obtain the following corollary.

Corollary 4.5 $\Pr(x - E(x) \geq \sqrt{n \log n}/8) \geq \sqrt{\frac{\log n}{n}}$

Define by p the number of living processes at the beginning of round r. Then for every processor i, we get $N_i^{r-1} \geq p$. We will look at the strategy of the adversary at each round. If the number of 1 messages sent in round r is less than $6p/10$, all processes either propose 0 or flip coins. So with probability higher than 1/2 more than 1/2 of the living processes will propose 0 for the next round. If all processes see at least one message of 0 in round $r+1$ than all processes will propose 0 for round $r+2$ (since they saw less than 1/2 messages of 1 and at least one message of 0), and the algorithm ends according to lemma 4.1. So in order to continue some of the processes must not see any message of 0 in round r (so they will have to propose 1 according to the algorithm no matter how many messages of 1 they so) and the adversary will have to fail

at least $p/2$ processes in round r and $r+1$. On the other hand, if less than $3p/10$ of the messages sent in round r are 0, the adversary will have to fail at least $p/10$ processes. If it wouldn't do this, all processes will propose 1 for the next round and again the algorithm ends according to lemma 4.1. So in order to keep the algorithm going (without failing a percent of the processes), more than $6p/10$ of the messages in round r must be 1, and more than $3p/10$ of the messages must be 0. Since the gap between proposing 0 and 1 is greater than $p/10$, it means that all the processes proposing 0 in round r are doing so due to a coin flip in the previous round (or else the adversary failed more than $p/10$ processes in one of the two previous rounds). So if the adversary didn't fail a percent of the processes in the two previous rounds, and the number of processes that flip coins in round $(r-1)$ is less than $6p/10$ the adversary will have to fail $p/10$ processes in round r with probability higher than $1/2$.
The next lemma uses these facts to estimate the expected number of 1 messages in round r. Denote by X the group of living processes at the end of round $(r-1)$, and by M the group of processes that flip coins in round $(r-1)$, set $k = |M|$.

Lemma 4.6 Denote by $E(x)$ the expected number of 1 messages in round r. Then if the adversary didn't fail $p/10$ processes in the two previous rounds and $E(x) < (\frac{6p}{10} + \frac{\sqrt{p\log p}}{16})$ the expected number of processes the adversary has to fail in this round is $\frac{p}{2}\frac{1}{2}\sqrt{\log p/p}$, or else the algorithm ends according to lemma 4.1.

Proof: As we showed, if $k < 6p/10$ the adversary has to fail more than $p/10$ processes with probability $> 1/2$ and we are finished. If there are less than $6p/10$ messages of 1 in round r the adversary has to fail $p/2$ processes with probability higher than $1/2$, so in order to prove the lemma it is enough to show that if $E(x) < (6p/10 + \sqrt{p\log p}/16)$ then the probability that less than $6p/10$ messages of 1 are sent in round r, is greater than $\sqrt{\log p/p}$. Let x be the random variable counting the number of 1 messages from the group X in round r. Let m be the same for the group M. Then $x - E(x) = m - E(m)$ since all the processes in $(X \setminus M)$ propose 1 without flipping a coin. Since $2k > p$ we get
$\Pr[m - E(m) \geq \sqrt{p\log p}/16] \geq \Pr[m - E(m) \geq \sqrt{2k\log 2k}/16] \geq \Pr[m - E(m) \geq \sqrt{k\log k}/8]$
and according to corollary 4.5 we get $\Pr[m - E(m) \geq \sqrt{k\log k}/8] \geq \sqrt{\log k/k} \geq \sqrt{\log p/p}$.
so $\Pr[x - E(x) \geq \sqrt{p\log p}/16] \geq \sqrt{\log p/p} \Rightarrow \Pr(x < 6p/10) \geq \sqrt{\log p/p}$. But if $x < 6p/10$ it means that the adversary will have to fail at least $p/2$ processes with probability $1/2$, so the expected number of processes the adversary has to fail is $\frac{p}{2}\frac{1}{2}\sqrt{\log p/p}$ proving the lemma. •

Using this lemma can show

Theorem 2 For any t, $t = \Omega(n)$ the algorithm SynRan reaches agreement in $O(t/\sqrt{n\log n})$ expected number of rounds.

Proof: Mark a round black if the expected number of 1 messages are less than $6p/10 + \sqrt{p\log p}/16$. Mark a round red if the expected number is higher.
Divide the run into blocks of 3 rounds. At each three rounds if one of the rounds is colored red, then the expected number of processes the adversary has to fail in this three rounds is higher than $\sqrt{p\log p}/16$. Otherwise all three rounds are marked black. if the adversary didn't fail $p/10$ processes in the first two rounds, than according to lemma 4.6 the expected number of processes it has to fail at the third is $\sqrt{p\log p}/4$. So in both cases the expected number of processes the adversary has to fail in each block is higher than $\sqrt{p\log p}/16$. If t is less than $n/2$, the expected number of rounds is less than $\frac{50t}{\sqrt{n/2\log n/2}}$ and the lemma is proved. Otherwise in order to get from n living processes to $n/2$ living processes, the expected number of rounds is less than $3\frac{n}{2}\frac{16}{\sqrt{n/2\log n/2}}$, and to get from $n/2^k$ to $n/2^{k+1}$ the expected number of rounds is less than $3\frac{n}{2^{k+1}}\frac{16}{\sqrt{n/2^{k+1}\log n/2^{k+1}}}$. So summing up over all k (until $n/2^k < \sqrt{n/\log n}$ when we move to the deterministic algorithm), we get that the expected number of blocks we perform the probabilistic algorithm is less than :

$\sum_{k=1}^{3\log n/4} \frac{16\frac{n}{2^k}}{\sqrt{\frac{n}{2^k}\log\frac{n}{2^k}}} = 16\sqrt{n}\sum_{k=1}^{3\log n/4} \frac{1}{2^{k/2}\sqrt{\log n - k}} \leq$
$32\sqrt{\frac{n}{\log n}}\sum_{k=1}^{3\log n/4} \frac{1}{2^{k/2}} \leq 100\sqrt{\frac{n}{\log n}}$. So if $t \geq n/2$ the expected number of rounds we perform the probabilistic algorithm is $O(\sqrt{n/\log n})$. Since the deterministic algorithm is performed at most $\sqrt{n/\log n}$ rounds the lemma is proved. ●

Since the algorithm SynRan has a finite expected number of round we know that it will terminate with probability 1.
The analysis above assumed $t = \Omega(n)$. For smaller t, let $r = 2 + t/\sqrt{n}$. Replacing $\sqrt{\log n}$ by $\sqrt{\log r}$ in our arguments our analysis shows that our algorithm and our lower bound adapted to this case together with lemma 4.1 and lemma 4.2 provide:

Theorem 3 For any $t < n$, SynRan guarantees Agreement, Validity, and Termination with probability 1. Moreover, the expected number of rounds to reach agreement is $\Theta(t/\sqrt{n\log(2 + t/\sqrt{n})})$.

References

[Asp97] J. Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pages 559–568. ACM, 1997.

[BO83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In Proceedings of the 2nd Annual ACM Symposium on the Principles of Distributed Computing, pages 27–30, 1983.

[BOL89] M. Ben-Or and N. Linial. Collective coin flipping. In Advances in Computing Research 5: Randomness and Computation, pages 91–115. JAI Press, 1989.

[CC85] B. Chor and B. Coan. A simple and efficient randomized byzantine agreement algorithm. IEEE Trans. on Software Engineering, SE-11(6):531–539, 1985.

[CD89] B. Chor and C. Dwork. Randomization in byzantine agreement. In Advances in Computing Research 5: Randomness and Computation, pages 443–497. JAI Press, 1989.

[CMS89] Benny Chor, Michael Merritt, and David B. Shmoys. Simple constant-time consensus protocols in realistic failure models. Journal of the ACM, 36(3):591–614, 1989.

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

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. SIAM Journal on Computing, 26, 1997.

[GM93] J. A. Garay and Y. Moses. Fully polynomial byzantine agreement in $t+1$ rounds. In Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pages 31–41. ACM, 1993.

[GP90] O. Goldreich and E. Petrank. The best of both worlds: Guaranteeing termination in fast randomized byzantine agreement protocols. Information Processing Letters, 36:45–49, 1990.

[Lin94] N. Linial. Game theoretic aspects of computing. In Handbook of Game Theory, volume 2, pages 1339–1395. Elsevier Science, 1994.

[Lyn96] Nancy A. Lynch. Distributed Algorithms. Morgan Kaufmann, 1996.

[Rab83] M. O. Rabin. Randomized byzantine generals. In Proceedings of the 24th Annual IEEE Symposium on the Foundations of Computer Science, pages 403–409. IEEE, 1983.

[Sch81] G. Schechtman. Lévy type inequality for a class of finite metric spaces. In Lecture Notes in Mathematics : Martingle Theory in Harmonic Analysis and Banach Spaces, volume 939, pages 211–215. Springer-Verlag, 1981.