Accurate Byzantine Agreement with Feedback

Vijay K. Garg*, John Bridgman and Bharath Balasubramanian
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712-1084, USA, Fax: 512 4715120
garg@ece.utexas.edu,{johnfb,bbharath}@mail.utexas.edu

Abstract

The standard Byzantine Agreement (BA) problem requires non-faulty processes to agree on a common value. In many real-world applications, it is important that the processes agree on the *correct* value rather than any value. In this paper, we present a problem called Accurate Byzantine Agreement (ABA) in which all processes get a common feedback (or payoff) from the environment indicating if the value they agreed upon was correct or not. The solution to this problem, referred to as the ABA algorithm, requires the non-faulty processes to incorporate the feedback so that their chance of choosing the correct value improves over subsequent iterations of the algorithm. We present an algorithm that solves the ABA problem based on two key ingredients: a standard solution to the BA problem and a multiplicative method to maintain and update process weights indicative of how often they are correct. We give guarantees on the accuracy of the algorithm based on assumptions on the accuracy of the processes and the proportion of faulty and non-faulty processes in the system. For each iteration, if the weight of accurate processes is at least $3/4^{th}$ the weight of the non-faulty processes, the algorithm always decides on the correct value. When the non-faulty processes are accurate with probability greater than $1/2$, the algorithm decides on the correct value with very high probability after some initial number of mistakes. In fact, among n processes, if there exists even *one* process which is accurate for all iterations, the algorithm is wrong only $O(\log n)$ times for any large number of iterations of the algorithm.

Keywords: Byzantine Agreement, Weighted Majority, Multiplicative Update

1 Introduction

In real-world applications, processes in a distributed system may be compromised, leading to malicious or arbitrary behavior. The Byzantine Agreement (BA) problem [PSL80, LSP82, FM97, DRS90, GM93] requires all non-faulty processes to agree on a common binary value given that some of the processes may show arbitrary faulty or Byzantine behavior. In the standard version of the problem, the value that is agreed upon may be either of the binary values so long as it is proposed by at least one non-faulty process. In some scenarios, it is better for the system to agree on a specific value among the two binary values. For example, suppose in a distributed control system a coordinated action needs to be taken (such as opening or closing a valve) depending upon the observations made by possibly faulty distributed processes. Depending upon the outcome of the action, the environment can provide a feedback if the action taken was correct or not. As another example, suppose that the system is making decision on whether to sell a stock based on recommendations made by multiple

*This research was supported in part by the NSF Grants CNS-0718990, CNS-0509024, Texas Education Board Grant 781, SRC Grant 2006-TJ-1426, and Cullen Trust for Higher Education Endowed Professorship.

processes. The final closing price of the stock provides a feedback for the decision made. Thus, the system or the environment can usually provide feedback to the non-faulty processes about which of the values was preferred or correct for that iteration of the agreement algorithm. Can the non-faulty processes use this feedback in a way that the probability of choosing the correct value increases in subsequent iterations of the algorithm?

We refer to this version of the BA problem as Accurate Byzantine Agreement (ABA) and define it as follows. Assume a set of n processes among which at most f Byzantine faults can occur. All non-faulty processes are required to make decisions for multiple rounds or iterations. For each iteration, a process can propose a binary value 0 or 1. All nonfaulty processes must agree on each decision and must take finite time to agree. After each decision, the environment provides a common feedback to all processes indicating if their decision was correct or wrong. The goal is to design an algorithm that maximizes the (expected) number of correct decisions by non-faulty processes over iterations of the algorithm.

In this paper, we give an algorithm, referred to as the ABA algorithm for the ABA problem. Our method relies on maintaining a common weight vector at all processes and updating this vector based on the feedback for each iteration. Initially, the weight of each process is a non-negative value proportional to the trust of the system on that process. If there is no prior information available, then the weights can simply be initialized to $1/n$. We use a weighted majority rule to determine the agreed upon value for the ABA problem. Once the value is committed, the feedback determines whether the decided value was a mistake or not. An important aspect of the algorithm is how the weights are updated based on the feedback. One possibility is to penalize all processes that proposed a wrong value after each iteration. Another possibility is to penalize processes only if the value decided in that iteration was wrong. Somewhat surprisingly, the behavior of the ABA algorithm may crucially depend upon which rule is used. We provide guarantees on the accuracy of the algorithm based on different assumptions on the accuracy of the processes and different weight update rules.

Byzantine Agreement is a well-studied problem in the field of distributed computing with research in both the theoretical [KS10, ADGH06, HM97, FM98] and practical aspects [CL99, CML$^+$06, CMW$^+$09]. For the synchronous model of communication (as assumed in this paper), it is known that agreement can be achieved only when $n \geq 3f + 1$ [PSL80]. In our work in [GB10], we present algorithms and bounds for weighted BA, where processes are assigned weights according to the application. In that paper, we give Byzantine agreement protocols that work even when $n < 3f + 1$, where f is the number of processes that have failed so long as the ratio of the weight of the failed processes to the weight of nonfaulty processes is at most $1/2$. We also present techniques to increase the weights of the non-faulty processes relative to that of the faulty processes based on detection of faulty behavior. Weighted BA problem does not have any notion of *accurate* value for agreement or environmental feeback as required for the ABA problem. It can be used as a subroutine in the ABA algorithm as shown in Section 5. Other approaches to BA include the use of artificial neural networks [WK01, LE07], randomized algorithms [Rab83, Bra87] or authentication based algorithms [DS83, PSL80]. None of these works explore the notion of accurate processes or the correct value for agreement. Our work can be applied to extend the results of these papers.

The concept of weighted majority and multiplicative weight update is used in many disciplines such as learning theory, game theory and linear programming [Kal07, LW94]. In the literature for this methodology, the experts are independent entities and there is no notion of liars that can collude and confuse other experts into suggesting the wrong value. In this paper, we assume the presence of malicious Byzantine experts and design algorithms to tolerate them. In summary, we make the following contributions:

- *The ABA Problem*: We introduce the problem of Accurate Byzantine Agreement, where the processes have to agree on a correct binary value as deemed by environmental feedback. The goal is to use this feedback to improve the accuracy of the algorithm in subsequent iterations.

- *The ABA Algorithm*: We present an algorithm to solve the ABA problem that uses a standard solution to the BA problem and a multiplicative method to maintain and update process weights. We make guarantees on the accuracy of the algorithm for the following models:
  - *Deterministic Accuracy*: We make assumptions on two ratios, the *accuracy ratio* (α) and the initial fault ratio (r_0). The accuracy ratio is the ratio of weight of the accurate processes to the weight of the non-faulty processes. The fault ratio r is the ratio of the weight of the faulty processes to that of the non-faulty processes. When $\alpha > 3/4$, the algorithm is always accurate if $r_0 < 1/2$. We relax this bound and show that when $\alpha > (1/2 + d)$, for any $0 \leq d \leq 1/2$, the algorithm is always accurate if $r_0 < 2d$.
  - *Probabilistic Accuracy*: We make assumptions on the probability with which non-faulty processes propose the correct value, β, and on the fault ratio r. When $\beta > 1/2 + d$ for any $0 < d < 1/2$, the probability of the algorithm being inaccurate is exponentially small if $r < 2d$.
  - *At-Least-One Accuracy*: If there exists at least one process such that it is inaccurate at most b times, then the ABA algorithm is inaccurate only $O(b + \log n)$ times. Hence, the algorithm tracks the most accurate process in the system.
- *Experimental Evaluation*: We present simulation results evaluating the performance of three distinct solutions: the ABA algorithm (with update on inaccuracy), the ABA algorithm with update on every iteration (always update) and the standard Byzantine Agreement (never update). While always-update and never-update perform very well for one of the models each, they perform poorly for the other one. The update on inaccuracy method performs well for both the models. The experimental results are presented in the appendix due to lack of space.

2 Model and Definitions

We consider a distributed system of n processes, $P_1 \ldots P_n$ with a completely connected topology. We assume that the underlying system is *synchronous* i.e., there is an upper bound on the message delay and on the duration of actions performed by processes. The communication system is assumed to be reliable and hence, no messages are dropped. The processes may undergo Byzantine failures, i.e., fail in an arbitrary fashion; in particular, they may lie and collude with other failed processes to foil any algorithm. However, they may not fake their identity.

We classify the processes in our system based on their behavior into non-faulty, accurate and faulty processes. While the notion of faulty and non-faulty processes is common to all BA problems, we introduce the concept of accurate processes that captures the idea of a correct proposal. A non-faulty process is considered *accurate* for an iteration if it proposes the correct value for that iteration.

In the standard BA problem, all non-faulty processes must agree on a common value. The only requirement on the decided value is that it must be proposed by a non-faulty process. In our proposal, the value decided by the algorithm is important as there is a reward function associated with the value decided, awarded by the environment or the system. The *correct* value is assigned 1 unit of reward and an incorrect value is assigned 0 units, i.e., no reward. Based on the reward, we replace the standard concept of validity with the notion of *accuracy*. Validity specifies that the value decided by the non-faulty processes must have been proposed by at least one of the non-faulty processes. This condition eliminates the trivial solution where all non-faulty processes agree on a fixed value all the time. In our system, the accuracy requirement eliminates the trivial solution. We define our problem below.

Definition 1 *(Accurate Byzantine Agreement with Feedback) Consider n processes consisting of non-faulty and faulty processes. There are multiple binary decisions that these n processes are required*

to make. For each possible decision (iteration of the ABA problem), each of the non-faulty processes proposes either 0 *or* 1*. An algorithm that solves the Accurate Byzantine Agreement with Feedback (ABA) problem, must guarantee the following properties:*

- *Agreement: For each iteration, all non-faulty processes decide on the same value.*
- *Termination: The algorithm terminates in a finite number of rounds.*
- *Accuracy: The non-faulty processes agree on a value that is deemed correct by environmental feedback. We define three distinct models of accuracy requirements:*
  - *Deterministic Accuracy requires that all non-faulty processes* always *decide on a value that is correct.*
  - *Probabilistic accuracy requires that the probability of making a mistake is small assuming that non-faulty processes propose the correct value with probability greater than* $1/2$*.*
  - *Cumulative accuracy requires that the total number of mistakes made by the algorithm is bounded assuming that there is at least one process that makes very few mistakes (also referred to as at-least-one accuracy).*

To incorporate the feedback provided by the environment we assign a non-negative weight w_i to each process P_i that provides an estimate, possibly erroneous, of the trust placed on that process. We summarize our notation in table 1.

Table 1: Notation

n	Number of processes	f	Number of Byzantine faults
w_i	Weight of process P_i	a	Total weight of accurate processes
p	Total weight of non-faulty processes	q	Total weight of faulty processes
r	Fault Ratio ($= q/p$)	α	Accuracy ratio ($= a/p$)

3 The ABA Algorithm

In this section, we propose an algorithm (Figure 1) for the ABA problem. The algorithm is identical at all processes and executes in synchronous iterations. At each process, we maintain two vectors W and V. Vector W stores the weight of each process while vector V stores the value proposed by each of them. Initially, the weight of each process is a non-negative value directly proportional to the initial trust on that process. In each iteration of the algorithm, non-faulty process proposes a value and executes Step 1 to Step 5 of the algorithm.

In Step 1, all processes exchange their proposed values to populate V. If no value is received from some process, the corresponding entry is set to 0. Since faulty processes may send conflicting values to other processes, it is not guaranteed that the V vector is identical at all non-faulty processes after Step 1.

In Step 2, the algorithm requires all non-faulty processes to agree on the value proposed by every other process and thereby make the V vector identical at all non-faulty processes after Step 2 of any iteration. For this step, we can use any standard BA algorithm such as the King algorithm [BGP89] that requires $n \geq 3f + 1$, or the Queen algorithm [BG89] that requires $n \geq 4f + 1$. The validity property satisfied by these algorithms ensures that the value of $V[i]$ for any non-faulty process P_i is exactly the value proposed by P_i.

In Step 3, processes determine the sum of weights of all processes that support value 0 or 1. The value with larger support, i.e., the weighted majority is chosen as the value in *decided*.

```
var
 W: array[1..n] of float initialized according to system trust (default value all 1/n);
 V: array[1..n] of {0, 1} initialized to 0;

 for iteration := 1 to t do
      V[i] = proposed value by P_i;

      // Step 1: Exchange values with all
      for j : 1 to n do
           send V[i] to P_j;
           receive V[j] from P_j; //if (no value received from P_j), V[j] = 0;

      // Step 2: Agree on V vector
           for j : 1 to n do: run standard Byzantine Agreement on V[j];

      // Step 3: Compute support for values 0 and 1 and choose the majority value
      float s0 = ∑_j{W[j] | V[j] = 0};     float s1 = ∑_j{W[j] | V[j] = 1};
      if (s0 ≥ s1) then decided := 0; else decided := 1;

      // Step 4: Wait for reward and determine the correct value based on the feedback
      if (reward = 1) then correctVal := decided;
      else correctVal := 1 − decided; //the process decided on the wrong value

      // Step 5: //multiplicative weight update on inaccuracy: ABA(UI)
      if (reward = 0) then
           for j : 1 to n do
                if (V[j] ≠ correctVal) then W[j] = (1 − ε) ∗ W[j];

      // Alternative Step 5': //multiplicative weight update on all iterations ABA(UA)
      for j : 1 to n do
           if (V[j] ≠ correctVal) then W[j] = (1 − ε) ∗ W[j];

 endfor;
```

Figure 1: The ABA Algorithm at P_i

In Step 4, processes receive the common feedback from the environment to determine the correct value.

In Step 5, we carry out the update of weights. If the value decided was incorrect, then the weights of the processes that proposed an incorrect value is reduced by some constant proportion ϵ ($0 < \epsilon < 1$) of its previous weight (multiplicative update). As an alternative to step 5, in step 5', we carry out the weight update on all iterations irrespective of the reward value. If we update weights only on inaccuracy, we refer to the algorithm as ABA(UI) ("update on inaccuracy"). If we update weights on all iterations, we refer to the algorithm as ABA(UA) ("update always"). We now prove that both the versions of the algorithm guarantee the agreement and termination property specified in definition 1 independent of the assumptions on accuracy.

Theorem 1 *(Agreement & Termination) Assuming $n \geq 3f + 1$, all iterations of the ABA algorithm*

guarantee agreement and termination.

Proof:
Agreement: We show that after Step 2 of every iteration, all non-faulty processes have identical W and V vectors. The proof is by induction on the iteration number. At the first iteration, the vector W is identical at all non-faulty processes by the initialization. Now assume that the vector W is identical at the beginning of any iteration i. Because all processes agree on vector V using Byzantine agreement, all non-faulty processes will have identical V after Step 2. This implies that all non-faulty processes will have identical values of $s0$, $s1$, and *decided* after step 3 because these variables depend only on W and V. Since the reward function is assumed to be common, all non-faulty processes will have identical value of *correctVal* and therefore will update W in an identical manner. The value decided depends only on W and V vectors and hence all non-faulty processes agree on the same value.

Termination: This is a synchronous algorithm which executes in finite number of rounds and hence, termination is satisfied trivially. ■

The ABA algorithm guarantees another useful property: if a nonfaulty process proposes an accurate value, then it can never be penalized. This property exploits the validity condition satisfied by the BA algorithm used in Step 2. A non-faulty process P_i will send the same value to all non-faulty processes. Therefore, all non-faulty processes will have identical $V[i]$ when they invoke the BA algorithm. Therefore, by validity of the BA algorithm, $V[i]$ at all non-faulty processes will be identical to the one proposed by P_i.

4 Accuracy Guarantees of the ABA Algorithm

In the previous section, we have shown that ABA algorithm guarantees agreement and termination. This section focuses on the accuracy guarantees the algorithm can provide based on varying assumptions about the accuracy of the processes in the system. Since standard Byzantine agreement is used in Step 2, in this section we assume that $n \geq 3f+1$, according to the lower bound for the BA problem [PSL80]. In Section 5, we consider the case when $n \geq 3f + 1$ does not hold.

4.1 Deterministic Accuracy

For deterministic accuracy, we make guarantees based on the accuracy ratio α (ratio of the weight of accurate processes to the weight of non-faulty processes) and the fault ratio of the system r (ratio of the weight of faulty processes to the weight of non-faulty processes). We show that if $\alpha > 3/4$ for each iteration and if the initial fault ratio $r_0 < 1/2$, then the algorithm guarantees accuracy. Then we relax this requirement and show that it is sufficient that $\alpha > (1/2 + d)$ for each iteration such that $r_0 < 2d$, to guarantee accuracy.

We first show that as long as $\alpha > 1/2$ for each iteration, r never increases if we update weights only on error. This enables us to make guarantees just based on the initial fault ratio of the system. The proof crucially depends on the fact that we update the weights of inaccurate processes only when the algorithm chooses the incorrect value.

Lemma 1 *(Non-Increasing Fault Ratio) For any iteration, if the accuracy ratio $\alpha > 1/2$, then the fault ratio r cannot increase after that iteration of the ABA(UI) algorithm.*

Proof: In the ABA(UI) algorithm, the weights of the processes changes only when the algorithm makes a mistake. Consider the weight of the non-faulty processes, p. Since $\alpha > 1/2$, when the algorithm makes a mistake, greater than $p/2$ of the weight will be unaffected and less than $p/2$ of the

weight will be reduced by a factor of $1 - \epsilon$. Hence, if p' is the weight of the non-faulty processes after a weight update,

$$p' > p/2 + (1 - \epsilon)p/2 = p(2 - \epsilon)/2 \quad (1)$$

Now consider the weight of the faulty processes q. The algorithm chooses the wrong value only when a majority weight, i.e. $> (p + q)/2$ of the weights are inaccurate. Since greater than $p/2$ of the weights are accurate, at least $q/2$ of the weights are inaccurate. Hence, if q' is the weight of the faulty processes after a weight update,

$$q' < q/2 + (1 - \epsilon)q/2 = q(2 - \epsilon)/2 \quad (2)$$

Dividing equation 2 by equation 1, we get, $q'/p' < q/p$. ■

Note that the proof for lemma 1 does not hold for the always update rule. If the faulty processes keep proposing the correct value, then the ABA(UA) algorithm will increase the relative weight of the faulty processes and consequently the fault ratio. If the fault ratio increases beyond 1 then Byzantine processes can force the ABA algorithm to choose incorrect value on crucial decisions.

In the following theorem we show that if $\alpha > 3/4$, then the ABA(UI) algorithm never makes a mistake as long as the initial fault ratio is less than $1/2$.

Lemma 2 *If the accuracy ratio $\alpha > 3/4$, and the initial fault ratio $r_0 < 1/2$, for all iterations, then the ABA(UI) algorithm guarantees accuracy.*

Proof: If the weight of accurate proposals is at least $3p/4$, then the weight of inaccurate proposals is at most $p/4 + q$. The algorithm selects the correct value if the accurate weight is more than the inaccurate weight. We need to show that, $p/4 + q < 3p/4$. Dividing both sides by p and rearranging, this is equivalent to showing that $r < 1/2$. Since $r_0 < 1/2$, from lemma 1, for all iterations, $r < 1/2$. ■

In the following theorem, we show that even if the accuracy ratio is just above $1/2$, the ABA algorithm never makes a mistake as long as the initial fault ratio is less than a certain threshold.

Theorem 2 *(Deterministic Accuracy) If the accuracy ratio $\alpha > 1/2 + d$ and if the initial fault ratio $r_0 < 2d$, for any $0 \leq d \leq 1/4$, for all iterations, then the ABA(UI) algorithm guarantees accuracy.*

Proof: If the weight of accurate proposals is at least $p(1/2 + d)$, then the weight of inaccurate proposals is at most $p(1/2 - d) + q$. The algorithm selects the correct value if the accurate weight is more than the inaccurate weight. Therefore, we need $p(1/2 - d) + q < p(1/2 + d)$. This condition is equivalent to $r < 2d$. Since $r_0 < 2d$, from lemma 1, for any iteration, $r < 2d$. ■

Note that when d equals $1/4$, this theorem reduces to lemma 2. Thus, theorem 2 generalizes lemma 2, when $d < 1/4$. Accuracy of the ABA(UI) is guaranteed if either an overwhelming majority of non-faulty processes is accurate (d is large) or there is a large percentage of non-faulty processes (r_0 is small).

The following theorem handles the case when a majority of the nonfaulty processes are accurate but the fault ratio is not smaller than $2d$.

Theorem 3 *(Accuracy after some initial mistakes) If the accuracy ratio $\alpha > 1/2 + d$, for any $0 \leq d \leq 1/4$, for all iterations, then the ABA(UI) algorithm guarantees accuracy after some initial mistakes.*

Proof: (Sketch) Similar to the proof of lemma 1, it is easy to show that there exists a constant γ such that the fault ratio decreases by a factor of at least γ for any mistake. Therefore, eventually the fault ratio becomes less than $2d$. Subsequently, by theorem 2 the algorithm ABA(UI) does not make any mistake. ■

It is easy to show that ABA(UA) algorithm can be forced to make unbounded mistakes by the Byzantine processes for any accuracy ratio less than $3/4$. Byzantine processes may initially propose correct values to increase the fault ratio. Once the fault ratio is high, they can ensure that ABA makes mistakes. They can repeat this cycle forever.

4.2 Probabilistic Accuracy

For probabilistic accuracy, we make guarantees based on the probability of accuracy of each non-faulty process β, and the fault ratio r, of the system. We show that if $\beta > 1/2 + d$ $(0 < d < 1/2)$, and $r < 2d$, the ABA algorithm guarantees accuracy with high probability.

Theorem 4 *(Probabilistic Accuracy) Let all weights in the system be in* $[0, 1]$*. If the accuracy probability of non-faulty processes* $\beta > 1/2 + d$ *and the fault ratio* $r < 2d$*, for any* $(0 < d < 1/2)$*, for all iterations, then the ABA algorithm guarantees accuracy with probability greater than* $1 - (\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}})^{\mu}$*, where* $\mu = p(1/2 + d)$ *and* $\delta = (2d - r)/(2d + 1)$*.*

Proof: Let X_i be the random variable indicating the non-faulty process P_i making an accurate proposal. Let $X = \sum_i w_i X_i$, where w_i is the weight of process P_i. We have $E[X_i] = 1/2 + d$. Therefore, $E[X] = (1/2 + d) \sum_i w_i = p(1/2 + d)$.

Let $\mu = E[X]$. We now show that $(1 - \delta)\mu = (p + q)/2$.
$(1 - \delta)\mu = (2d + 1 - (2d - r))/(2d + 1) * p * (2d + 1)/2 = (1 + r)p/2 = (1 + q/p)p/2 = (p + q)/2$.

When $r < 2d \leq 1$, we get that $0 < \delta < 1$. Hence, from Chernoff's bound, we have,
Pr[ABA algorithm makes wrong decision]
$= Pr$[sum of all weights supporting the correct value $< (p + q)/2)$] { From the algorithm }
$\leq Pr[X < (p + q)/2)]$ { Considering only non-faulty processes }
$= Pr[X < (1 - \delta)\mu]$ { Shown above}
$< (\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}})^{\mu}$, { From Chernoff's bound and $0 < \delta < 1$ }. ■

In Theorem 4, the error probability depends upon $\delta = (2d-r)/(2d+1)$. As r decreases, δ increases. We now show that for the ABA(UA) algorithm the ratio r is expected to decrease exponentially with increasing iterations.

Theorem 5 *(ABA(UA): Exponentially Decreasing Expected Fault Ratio) If the accuracy probability of non-faulty processes is at least* $1/2 + d$*, and the accuracy probability of faulty processes is at most* $1/2 - d$*, then there exists* $k > 1$*, such that after* j *iterations of the ABA(UA) algorithm, the expected ratio of the weight of the non-faulty processes to the weight of the faulty processes is at least* k^j/r_0*.*

Proof: We first show a bound on the expected weight of a non-faulty process after j iterations. Let the initial weight of a nonfaulty process be w_0. Let M_i be the random variable denoting the multiplicative factor at iteration i for a non-faulty process. Let W_j be the random variable denoting weight of a non-faulty process after j iterations. It is clear that for ABA(UA) algorithm, $W_j = w_0 \Pi_{i=1}^{i=j} M_i$. The multiplicative factor for any iteration depends on the environmental feedback and is independent of other iterations. Hence, $E[W_j] = w_0 \Pi_{i=1}^{i=j} E[M_i] \geq w_0 \Pi_{i=1}^{i=j} (1/2 + d) * 1 + (1/2 - d) * (1 - \epsilon) = w_0(1 - \epsilon/2 + d\epsilon)^j$.

Similarly, since the probability that a faulty process makes a correct proposal is at most $1/2 - d$, the expected weight of a faulty process after j iterations of the ABA(UA) algorithm is at most $(1 - \epsilon/2 - d\epsilon)^j$ times its original weight.

We now show that the expected fault ratio decreases exponentially with the number of iterations. Let p_0 and q_0 be the intial weights of non-faulty and faulty processes such that $q_0/p_0 = r_0$. Let S_j and T_j be the random variables to denote weights of the non-faulty processes and faulty processes after j iterations of the ABA(UA) algorithm. Since the expected weight of each non-faulty process after j iterations is at least $(1 - \epsilon/2 + d\epsilon)^j$ times its original weight; by linearity of expectation, $E[S_j] \geq p_0 * (1 - \epsilon/2 + d\epsilon)^j$. Similarly, $E[T_j] \leq q_0 * (1 - \epsilon/2 - d\epsilon)^j$. We now bound $E[S_j/T_j]$. Using independence of S_j and T_j, we get that $E[S_j/T_j] = E[S_j] * E[1/T_j]$. We now use the fact that for any nonnegative random variable X, $E[1/X] \geq 1/E[X]$ which can be shown using Jensen's inequality ($E[f(X)] \geq f(E[X])$ for convex f). Therefore,

$E[S_j] * E[1/T_j] \geq E[S_j] * 1/E[T_j]$
$\geq \frac{p_0*(1-\epsilon/2+d\epsilon)^j}{q_0*(1-\epsilon/2-d\epsilon)^j} = 1/r_0 * (1 + \frac{2d\epsilon}{1-\epsilon/2-d\epsilon})^j$

By defining $k = (1 + \frac{2d\epsilon}{1-\epsilon/2-d\epsilon})$, we get the desired result. Because $0 < \epsilon < 1$ and $0 < d < 1/2$, $(1 - \epsilon/2 - d\epsilon)$ is guaranteed to be positive which ensures $k > 1$. ■

Remark: The above theorem can be generalized to the case when non-faulty processes are accurate with probability at least $1/2 + d_1$ and faulty processes are accurate with probability at most $1/2 - d_2$. In this case, $k = (1 + \frac{\epsilon(d_1+d_2)}{1-\epsilon/2-d_2\epsilon})$. When $d_1 = d_2$ we get the original value of k. Also when $d_2 = -d_1$ (faulty processes are as accurate as non-faulty processes), we get k equals 1.

4.3 At-Least-One Accuracy

For this section, we assume that there is at least one process in the system that is inaccurate *only* for a small number of iterations of the ABA algorithm. This assumption is sufficient to guarantee *cumulative accuracy*, i.e., a bound on the total number of mistakes made by the algorithm. Our results are based on the method of weighted majority with multiplicative updates [Kal07]. We first consider ABA(UI) algorithm. In the following theorem, we show that ABA(UI) guarantees accuracy for a large number of iterations.

Theorem 6 *(At-Least-One Accuracy, ABA(UI)) Assume $n \geq 3f+1$. If there exists at least one process such that is inaccurate at most b out of j iterations of the ABA(UI) algorithm, then the algorithm is inaccurate at most $2(1 + \epsilon)b + (2/\epsilon) \log n$ times.*

Proof: The proof follows from standard arguments in multiplicative update method [Kal07]. We assume that the process weights are all initialized to $1/n$. Let $\phi(i)$ be the sum of all the weights of the processes at the end of iteration i. Suppose that for any iteration i, the ABA(UI) algorithm is wrong. This means that the weighted majority of the values in the proposed vector were wrong and hence a majority of the weights will decrease by $(1 - \epsilon)$ of their previous value. Therefore, $\phi(i) \leq \phi(i-1)/2 + \phi(i-1)/2 * (1 - \epsilon) = \phi(i-1)(1 - \epsilon/2)$. The total weight, at the beginning of the algorithm, $\phi(0)$ is equal to one. Suppose that the ABA(UI) algorithm makes $m(j)$ mistakes in the first j iterations. After j iterations of ABA(UI), we get $\phi(j) \leq \phi(0)(1 - \epsilon/2)^{m(j)} = (1 - \epsilon/2)^{m(j)}$.

Now consider a nonfaulty process that is inaccurate at most b out of j iterations. In spite of the presence of Byzantine processes, ABA(UI) algorithm guarantees that this process is never penalized when it is accurate. The weight of this process is at least $(1 - \epsilon)^b$*(its initial weight) $= (1 - \epsilon)^b/n$. This weight is less than the total weight. Therefore, $(1 - \epsilon)^b/n < (1 - \epsilon/2)^{m(j)}$.

Taking log on both sides and shifting n to the right hand side, we get $b \log(1 - \epsilon) < \log n + m(j) \log(1 - \epsilon/2)$.

Dividing both sides by $\log(1 - \epsilon/2)$ which is a negative quantity and rearranging gives us

$$m(j) < b \log(1 - \epsilon)/\log(1 - \epsilon/2) - \log n/\log(1 - \epsilon/2).$$

In the following part of the proof, we use two inequalities: $-\log(1-\epsilon) \leq \epsilon+\epsilon^2$ and $-\log(1-\epsilon/2) \geq \epsilon/2$ that require $\epsilon < 0.684$. Applying these inequalities we get, $m(j) < b * 2 * (\epsilon + \epsilon^2)/\epsilon + 2 \log n/\epsilon$.

Therefore, $m(j) < 2(1 + \epsilon)b + 2/\epsilon \log n$. ■

Interestingly, the result holds even when we use ABA(UA).

Theorem 7 *(At-Least-One Accuracy, ABA(UA)) Assume $n \geq 3f + 1$. If there exists at least one process such that is inaccurate at most b out of j iterations of the ABA(UA) algorithm, then the algorithm is inaccurate at most $2(1 + \epsilon)b + (2\epsilon) \log n$ times.*

Proof: Note that even when we update weights on all iterations, the following inequalities hold. The total weight in the system, $\phi(j) \leq \phi(0)(1-\epsilon/2)^{m(j)} = (1-\epsilon/2)^{m(j)}$. The weight of the process that is wrong b out of j iterations is $(1-\epsilon)^b *$(its initial weight) $= (1-\epsilon)^b/n$. Hence, the previous proof applies. ■

Substituting $b = 0$ in the above theorem, i.e., when at least one process is accurate for all j iterations of the algorithm, the ABA algorithm makes a mistake only $O(\log n)$ times. Note that this is independent of the number of iterations and hence, if the ABA algorithm is run for a large number of iterations ($j >> \log n$), then it guarantees accuracy in most of them. Or in other words, the ABA algorithm is approximately as accurate as the most accurate process in the system.

5 ABA Algorithm with Weighted Byzantine Agreement

In the ABA algorithm proposed in Figure 1, we have used standard Byzantine Agreement in Step 2. Since standard Byzantine Agreement assumes $n \geq 3f+1$, the ABA algorithm also made the same assumption. This assumption is crucial for correctness of the ABA algorithm because agreement requires that processes have identical V vector after step 2. We now consider the case when $n < 3f+1$, but the initial fault ratio is less than $1/2$. Thus, more than a third of the processes may be faulty but the total weight of the faulty processes is still less than $1/2$ of the weight of the nonfaulty processes. Under this scenario, we propose an alternative to ABA algorithm by replacing Step 2 of the ABA algorithm by

// Step 2'(Alternative to Step 2) : Agree on V vector

for j : 1 to n do: run *Weighted* Byzantine Agreement [GB10] on $V[j]$

Thus, we use the weight vector even to agree on the value of $V[j]$ (as used by the algorithms in [GB10]). We refer to this algorithm as ABAW algorithm. Since the fault ratio decreases under various accuracy assumptions, the ABAW algorithm works correctly even when the set of processes that act Byzantine increases with time so long as the fault ratio stays less than $1/2$. The following theorem can be shown for the ABAW algorithm analogous to that for ABA algorithm.

Theorem 8 *Assuming $r_0 < 1/2$, all iterations of the ABAW algorithm guarantee agreement and termination if the weight update method ensures $r < 1/2$.*

For the deterministic accuracy property, we have the following theorem.

Theorem 9 *If the accuracy ratio $\alpha > 1/2+d$ and if the initial fault ratio $r_0 < 2d$, for any $0 \leq d \leq 1/4$, for all iterations, then the ABAW(UI) algorithm guarantees accuracy.*

Note that Theorem 3 does not hold for ABAW(UI) because we require $r_0 < 1/2$. Theorem 4 holds for ABAW(UA) without the assumption of $n \geq 3f+1$ (assuming $r < 1/2$). Theorem 6 does not hold for ABAW(UI) or ABAW(UA) because the fault ratio may increase beyond $1/2$ even if one process is accurate most of the times.

6 Conclusion and Future Work

We introduce the problem of Accurate Byzantine Agreement with Feedback where in addition to agreeing on the same value, the processes in the system have to agree on the correct value. The notion of correctness is based on the environment or any kind of external feedback common to all the processes in the system. We present an algorithm that solves the problem for various assumptions on the initial accuracy and weight distribution of the processes. We show that if the weight of the

accurate processes is greater than $3/4$ the weight of the non-faulty processes, then the algorithm always decides on the correct value. We relax this further and show that if a majority of the non-faulty processes are accurate, then for certain assumptions on the faulty and non-faulty processes, the algorithm never makes a mistake. Further, we show that if the probability of accuracy of the non-faulty process is greater than $1/2$, then the algorithm's accuracy improves exponentially in the number of mistakes it makes. Finally, we consider the simple assumption that at least one process always proposes the correct value for all iterations and show that the algorithm rarely makes mistakes.

We performed simulations comparing the performance of three different weight update methods: update on inaccuracy, always update and never update (just standard Byzantine agreement). The experiments compared the performance of these solutions under all three accuracy assumptions and the results indicate that while never update and always update perform very well for different fault models, update on inaccuracy performs uniformly well for both fault models.

This problem brings up many further questions. The results in this paper mainly present upper bounds for the problem of accurate Byzantine agreement with feedback. We also need to explore lower bounds for the problem. Also, our results depend on the multiplicative update rule. We wish to explore other update rules, such as additive updates and compare their performance, both theoretically and practically. Designing optimal policies to guarantee maximum probability of correct decisions is also an interesting problem.

References

[ADGH06] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, PODC '06, pages 53–62, New York, NY, USA, 2006. ACM.

[BG89] Piotr Berman and Juan A. Garay. Asymptotically optimal distributed consensus (extended abstract). In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, pages 80–94, London, UK, 1989. Springer-Verlag.

[BGP89] P. Berman, J.A. Garay, and K.J. Perry. Towards optimal distributed consensus. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:410–415, 1989.

[Bra87] Gabriel Bracha. An $O(\log n)$ expected rounds randomized Byzantine generals protocol. *Journal of the ACM*, 34(4):910–920, October 1987.

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Third Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, February 1999. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS.

[CML+06] James Cowling, Daniel Myers, Barbara Liskov, Rodrigo Rodrigues, and Liuba Shrira. Hq replication: A hybrid quorum protocol for byzantine fault tolerance. In *Proceedings of the Seventh Symposium on Operating Systems Design and Implementations (OSDI)*, Seattle, Washington, November 2006.

[CMW+09] A. Clement, M. Marchetti, E. Wong, L. Alvisi, and M. Dahlin. Making byzantine fault tolerant systems tolerate byzantine faults. In *6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2009.

[DRS90] Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[FM98] Matthias Fitzi and Ueli M. Maurer. Efficient byzantine agreement secure against general adversaries. In *DISC '98: Proceedings of the 12th International Symposium on Distributed Computing*, pages 134–148, London, UK, 1998. Springer-Verlag.

[GB10] Vijay Garg and John Bridgman. A report on the weighted byzantine agreement problem (to appear in proceedings of ipdps 2011). Technical Report TR-PDS-2010-002 `http://maple.ece.utexas.edu/TechReports/2010/TR-PDS-2010-002.pdf`, Parallel and Distributed Systems Laboratory, The University of Texas at Austin, 2010.

[GM93] Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement in t + 1 rounds. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 31–41, New York, NY, USA, 1993. ACM.

[HM97] Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 25–34, New York, NY, USA, 1997. ACM.

[Kal07] Satyen Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, Princeton, NJ, USA, 2007. AAI3286120.

[KS10] Valerie King and Jared Saia. Breaking the o(n2) bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '10, pages 420–429, New York, NY, USA, 2010. ACM.

[LE07] Kok-Wah Lee and Hong-Tat Ewe. Performance study of byzantine agreement protocol with artificial neural network. *Inf. Sci.*, 177(21):4785–4798, 2007.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, July 1982.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108:212–261, February 1994.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[Rab83] Michael O. Rabin. Randomized byzantine generals. In *Foundations of Computer Science*, pages 403–409, 1983.

[WK01] S. C. Wang and S. H. Kao. A new approach for byzantine agreement. In *Proceedings of the The International Conference on Information Networking*, page 518, Washington, DC, USA, 2001. IEEE Computer Society.

7 Appendix: Experimental Evaluation of ABA Algorithm

The experimental evaluation compares three different update methods: "always update", "update on inaccuracy" and "never update". The last option reduces to standard Byzantine agreement. The performance of the three accuracy models presented in this paper are considered with each of these update methods for two different Byzantine fault models. Always update and never update perform very well under one of the fault models each, while they both perform very poorly for the other. Update on inaccuracy, the method followed in this paper, is always close to the best.

7.1 Experimental Setup and Parameters

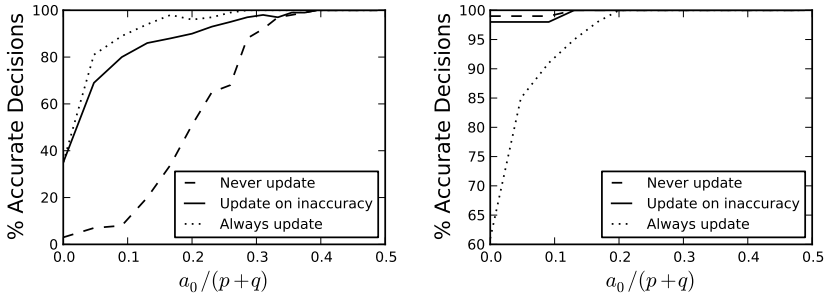
For the experimental evaluation, we focus on faulty processes that will always try to make the system agree upon an incorrect value. The faulty processes have complete knowledge of the system including the correct value for each iteration. Our simulation uses two models for faulty processes. Model 1 uses a process that will always propose the incorrect value. Model 2 uses a process that looks at the percentage of its own weight to the weight of all processes and proposes the correct value if its percentage is below a threshold and the incorrect value otherwise. There are two types of non-faulty processes used. The first is an accurate non-faulty process that always proposes the correct value ($d = 0.5, \beta = 1$). The second type of non-faulty process chooses the correct value with probability $\beta = 0.5 + d$, where $d \in [0, 0.5]$. The Queen algorithm [BG89] is used for step 2 in the ABA algorithm and for all simulations, $n = 41$, $f = 10$ and $\epsilon = 0.1$.

7.2 Results

Simulation results for *deterministic accuracy* are shown in Figure 2. For this experiment, we had one accurate process, and the other non-faulty processes had a value of $d = 0.00001$. We compare the % of accurate decisions made by the algorithm for 100 iterations, with increasing values of $a_0/(p + q)$ i.e. the starting weights of the accurate processes divided by the total weight of processes in the system. The experiments were performed for the two fault models 1 and 2. As can be seen, having an update method performs much better than not having one with model 1 and always updating performs poorly with model 2. Update on error gives a good compromise between the two.

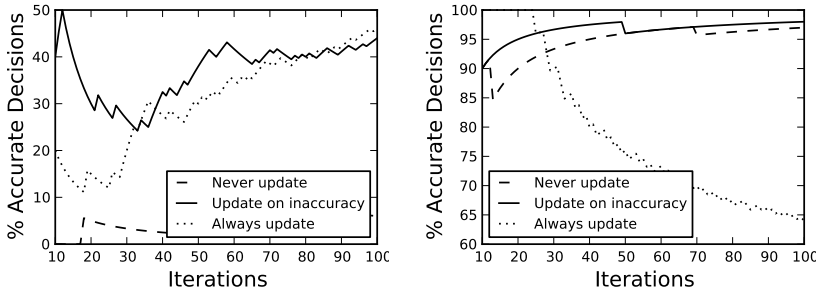
Results for *probabilistic accuracy* are shown in Figure 3. For this experiment, all non-faulty processes had $d = 0.02$ and all processes start with uniform weights. We compare the % of accurate decisions with increasing number of iterations. Notice that, on the whole, update on inaccuracy performs the best for these graphs. Always updating seems like the natural method to use but in Figure 3(b) always update performs the worst.

Simulation results for *at-least-one accuracy* are shown in Figure 4. For this experiment, we had one non-faulty process which is always accurate i.e. $d = 0.5$, and the remaining non-faulty processes had $d = 0.00001$. The processes start with uniform weights. We compare the % of accurate decisions with increasing number of iterations. For model 1 in Figure 4(a), updating weights increases the accuracy over iterations. With model 2 always update shows the worse performance with update on accurate being the best. Notice how update on inaccuracy is always close to the best.



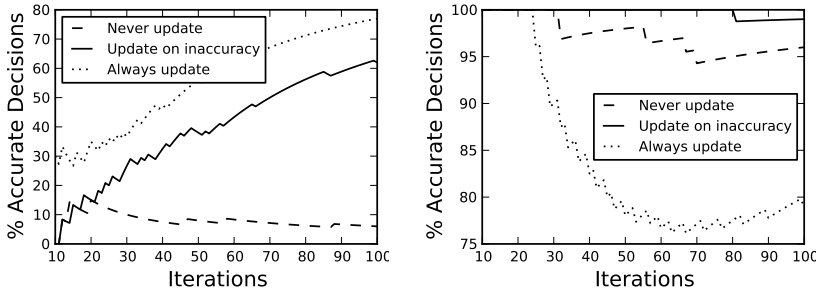
(a) With model 1. (b) With model 2.

Figure 2: Deterministic accuracy: Ratio of Accurate Process Weights vs. % Accurate Decisions



(a) With model 1. (b) With model 2.

Figure 3: Probabilistic Accuracy: Iterations vs. % Accurate Decisions, $d = 0.02$



(a) With model 1. (b) With model 2.

Figure 4: At-Least One Accuracy: Iterations vs. % Accurate Decisions, One accurate process