Fault Tolerant Weighted Voting Algorithms

Azad Azadmanesh[1], Alireza Farahani[2], and Lotfi Najjar[3]
(Corresponding author: Azad Azadmanesh)

Department of Computer Science, University of Nebraska, Omaha, 68182, USA[1]
(Email: azad@unomaha.edu)
Department of Computer Science & Information Systems, National University, San Diego, 92037, USA[2]
Department of Information System & Quantitative Analysis, University of Nebraska, Omaha, 68182, USA[3]
(Received Dec. 21, 2006; revised and accepted Apr. 23, 2007)

Abstract

Agreement algorithms can be categorized in different ways. One categorization of such algorithms is based on whether the final decisions are exact or inexact. In *inexact algorithms*, also known as *approximate agreement algorithms*, each node produces its final decision that may not be necessarily the exact same decision value produced by a different non-faulty node. Most studies on such algorithms are either oblivious to the confidence level placed on each node or the impact of malicious behavior is not accounted for. This study introduces a family of inexact agreement algorithms taking into account both the confidence level placed on each node and the presence of malicious behavior. Expressions are developed for the convergence rate and fault tolerance of these algorithms, and the effect of weights are shown when the agreement process favors nodes with a specific level of trust. The study also describes the difficulties for applying weights to the existing voting algorithms.

Keywords: Approximate agreement, Byzantine, data fusion, network security, sensor networks

1 Introduction

Voting algorithms have been used in many domains such as political elections, neural networks, sensor networks, and distributed systems. Voting algorithms, often called agreement algorithms, specify how the votes are integrated leading to the final results. They are used to create consistency and to mask out the effect of faulty behavior. The votes are the input values provided by multiple sources[1], and the final result that can be in either a scalar or in a vector form is referred to by many phrases such as the "decision value", "agreed value", "voted value", or simply the "final decision".

A simple example of an algorithm that produces the final result in a scalar form is a bit-voter based on the majority rule. The bit-voter receives the input values as a mixture of 0s and 1s, and outputs a 1 if the majority of input values is a 1, otherwise a 0. An example of an agreement algorithm that produces a vector result is in a fault tolerant software environment where multiple versions of a program are run on different platforms. The programs might produce conflicting results but all within the acceptable range of accuracy.

As some of the input values might be erroneous, the fusion of data in an appropriate way becomes of paramount importance vis-a-vis performance of the agreement algorithms and the impact of errors or misbehavior. The agreement algorithms can be categorized in different ways. One categorization is based on whether the final decision is *exact* or *inexact*. An exact value is in reference to agreeing on the same value by every non-faulty node [6]. An inexact agreement algorithm decides on real values that may not be a subset of the input values or necessarily be the same, but are within the acceptable range. The focus of this research is based on inexact algorithms, also known as *Approximate Agreement* (AA) [9].

In AA, each node decides on a final value based on its own input and the input values received from other nodes. The decision values, one from each node, may not be the same due to many conditions such as round-off errors, variation in message transmission times, or because of faulty nodes transmitting conflicting values to other nodes. AA algorithms have been used in the synchronization of distributed clocks that can drift from each others. They are also used in the growing field of sensor networks, where raw data from sensors are collectively analyzed producing greater precision about an object or an event [7, 8, 16]. Sensors have limited accuracy and many applications require them to be placed in hostile environments. Therefore, the design of an appropriate fault tolerant voting algorithm is vital. Some applications of sensor networks are in military surveillance, flight control systems, robotics, nuclear reactor systems, target trajectory and detection.

Given an arbitrarily small positive real value ϵ, an AA

[1]Depending on application, a source might be referred to as a "node","process", "host", or a "peer". In this article, these terms are used interchangeably.

algorithm must satisfy two conditions [9, 12]:

- *Agreement* – The decision values of any pair of non-faulty hosts are within ϵ of each other.
- *Validity* – The decision value for each non-faulty host is within the range of the initial correct values.

Most agreement algorithms follow rounds of message exchange. In AA algorithms, each non-faulty node executes the same voting algorithm in rounds as follows. Each node broadcasts its initial value to all nodes including itself. Each node then collects the values it has received into a multiset and according to some a priori-known rules the multiset is filtered out to produce a *voting multiset*. The node will then apply a function F to the voting multiset to attain its latest estimate of the final value to be used in the next round of message exchange. The AA conditions are achieved if every voting round is single-step convergent, i.e. the voted value by each non-faulty node is in the range of correct values and the diameter of voted values among the correct nodes shrinks in each round.

Approximate agreement algorithms in general are different from each other in the filtration aspect of the values received in each round. In addition, these algorithms do not distinguish among hosts in terms of amount of trust placed upon them. Having a different weight of trust on each host makes the process of obtaining the performance expressions for the existing algorithms very complex if not impossible, as will be explained in a later section. Therefore, it becomes necessary to introduce a new family of weighted voting algorithms. These algorithms employ the appropriate filtration step in such a way to ensure convergence (agreement) will take place while being able to measure the performance of these algorithms in the worst case possible.

Section 2 provides some research background on weighted voting. Section 3 explains the general voting process for the new family of weighted voting algorithms, and obtains the convergence rate and fault tolerance for such algorithms. Section 3.5 presents a working example on obtaining the worst convergence rate. Section 4 deals with the optimal performance of the weighted voting algorithms and Section 5 remarks on the number of nodes required to reach a desired level of performance. Section 6 compares these algorithms against the existing AA algorithms. Finally, Section 7 concludes the paper with a summary and some directions for future research.

2 Background

Weights have been used in different fields and applications with the goal of reconciling the domain values to a final, agreeable decision. Wu and Chen in [22] use the weights on k-out-of-n systems, where a system is said to be operating correctly if the cumulative sum of weights is at least k. Tong and Kain [21] employ votes as weights that are assigned to system components in such a way to maximize reliability. Other studies such as in [1, 10] utilize vote assignments for manipulating replicated data. Specifically, a replicated file is assigned a number of votes, and the transaction coordinator needs to reach a quorum by collecting a certain number of votes for a read or write operation, to successfully finish the transaction.

Parhami in [17] defines a weighted inexact voting algorithm, where a decision is made on a set of real numbers with their associated vote tallies. The algorithm requires the maximal set of numbers with the property that the threshold difference between each pair of the numbers is not greater than a small positive value ϵ. Then an averaging function is applied on the maximal set to reach a final decision. The averaging function can be specified in different ways, such as selecting the value with the highest number of votes in the maximal set. These algorithms, however, do not consider malicious behavior or necessarily guarantee the AA conditions.

With respect to treatment of values in the voting process, AA algorithms can in general be partitioned into different families: Oblivious, Egocentric, and Egophobic. These algorithms place the same level of trust on each host. Basically, if n values are voted upon, the trust weight for each node is $1/n$.

The family of oblivious algorithms do not place any preference on any input value. In other words, such algorithms are oblivious to the state or source of input values [12]. The Egocentric and Egophobic voting algorithms are normally used for applications where the diameter of correct values is known, but further synchronization is needed to prevent divergence in the future, or to provide more coordinated results among nodes. In Egocentric algorithms, a node favors values that are closer to its own or using its own value when possible [5, 14, 15, 18]. On the other hand, a node using an Egophobic algorithm places favoritism on values that are further away from its own. Such algorithms have been used for hardware clock synchronization [3, 11, 13, 19].

The research herein introduces a new family of non-oblivious algorithms with respect to a confidence level placed on each node. The new family of algorithms is called *Weighted Approximate Agreement* (WAA). Each member of this family attempts to reach a final decision closer to the values with higher weight of confidence. The study obtains the convergence rate and fault tolerance expressions. The WAA algorithms not only provide decision values based on confidence levels, but also guarantee that the decided values are among the range of correct values in the presence of faults.

3 Weighted AA (WAA)

Assume the network consists of N nodes. Further assume that the network is completely connected and the communication system is *synchronous*. In synchronous communication, the processing time and communication delay among nodes are bounded. Therefore, any data item received beyond the allocated period of time is considered erroneous.

The fault model considered consists of asymmetric and benign faults. Benign faults are self-incriminating or self-evident to "every" host. Examples of benign faults include out of bound data, or data received outside of the bounded communication delay. The maximum number of benign faults is assumed to be b. An asymmetric fault occurs when multiple hosts receive conflicting information from the same transmission source, due to faults in the communication network or the sending host maliciously transmits conflicting values to other hosts. The maximum number of asymmetric faults is assumed to be a.

Normally a large percentage of faults are benign and since these faults must be globally self-evident to all non-faulty nodes, every non-fault node can safely remove them from further consideration. Therefore, it is only necessary to address the agreement process using multisets of values not containing benign errors [12]. Hence, the size of the voting multiset is $n = N - b$, and the total number of faults is $t = a + b$.

3.1 Voting Process of WAA Algorithms

The WAA algorithms contain the following steps for each round of the voting process:

1) *Broadcast* – Each host i broadcasts its current value to every host including itself.

2) *Collect* – Each host i collects the current values broadcast by other hosts including its own value into a multiset $\mathbf{N}_i$.

3) *Sample* – Each host i filters $\mathbf{N}_i$ producing the voting multiset $\mathbf{V}_i$. Host i removes the benign faults from $\mathbf{N}_i$. If $\mathbf{N}_i$ does not contain a value from a specific host, host i will use its own value as the default value to be used in $\mathbf{V}_i$. Host i will also replace the highest a and the lowest a values in $\mathbf{N}_i$ with its own value. By replacing the extreme elements, the range of $\mathbf{V}_i$ will be within the range of correct values, so that the validity condition can be satisfied. Therefore, $|\mathbf{V}_i| = n$.

4) *Execute* – During this step, the approximation-function F is applied to generate a single voted value.

Consider a weight multiset $\mathbf{W} = \langle w_1, \ldots, w_n \rangle$, where w_i is a positive trust-weight placed on node i and the sum of weights is equal to one. Also consider a non-faulty node i has the voting multiset $\mathbf{V}_i = \langle v_{i,1}, \ldots, v_{i,n} \rangle$. WAA algorithms utilize the common weighted averaging function.

$$F(\mathbf{V}_i) = \sum_{g=1}^{n} w_g v_{i,g}. \tag{1}$$

Some approximation functions use a selection function [4, 5, 12]. A selection function allows a certain number of elements of $\mathbf{V}_i$ to participate in the evaluation of voted value in each round. At the end of Subsection 3.2, it will be shown how $F(\mathbf{V}_i)$ can be modified to include a selection function.

3.2 Convergence Rate

Before describing the voting process formally, define the following:

$\mathbf{U}_{all}$ = The multiset of all correct values received by non-faulty nodes.

$\mathbf{U}_{voted}$ = The multiset of voted values by non-faulty nodes, i.e. $\mathbf{U}_{voted} = \langle F(\mathbf{V}_1), \ldots, F(\mathbf{V}_n) \rangle$.

$\rho(\mathbf{V}_i)$ = The range of multiset $\mathbf{V}_i$, i.e. $[\min(\mathbf{V}_i), \max(\mathbf{V}_i)]$.

$\delta(\mathbf{V}_i)$ = The diameter of multiset $\mathbf{V}_i$, i.e. the difference between the maximum and minimum values of $\mathbf{V}_i$.

A voting algorithm is single-step-convergent if both of the following conditions are true following every round of voting:

- *Convergence* – For each pair of non-faulty nodes, i and j, the difference between their decision values is *strictly less than* the diameter of the multiset of correct values received, i.e. $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| < \delta(\mathbf{U}_{all})$.
- *Validity* – For each non-faulty node i, the decision value is within the range of correct values, i.e. $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{all})$.

The effectiveness of a convergent voting algorithm is measured by its convergence rate C. Assuming that $\delta(\mathbf{U}_{all}) > 0$, C is the ratio:

$$C = \frac{\delta(\mathbf{U}_{voted})}{\delta(\mathbf{U}_{all})}. \tag{2}$$

If $C < 1$ in each round, given enough number of rounds, it is then guaranteed that the system will achieve the Agreement condition. To obtain C, the maximum diameter of voted values in a round must be determined. Therefore, it is necessary to find the conditions under which $F(\mathbf{V})$ is maximized and minimized. To better understand these conditions, the following provides a simple example, which will be used as the basis for presenting these conditions formally.

3.3 Working Example 1

Consider a system with $n = 20$, $a = 5$, $\max(\mathbf{U}_{all}) = \top$, and a node i holding the value α. Let $\mathbf{M}_i$ be the filtered multiset right before replacing the extreme values. Sort $\mathbf{M}_i$ and resequence its corresponding weight multiset. Assume $\mathbf{M}_i$ can be partitioned into $\mathbf{U}_{all}$ and error multiset

$\mathbf{X}$ such that $x_k \geq \top$, $k \in \{1, \dots, a\}$. Accordingly,

$$\begin{aligned} \mathbf{U}_{all} &= \langle u_1, \dots, u_{15} \rangle \\ \mathbf{X} &= \langle x_1, \dots, x_5 \rangle \\ \mathbf{M}_i &= \langle u_1, \dots, u_{15}, x_1, \dots, x_5 \rangle \\ \mathbf{V}_i &= \langle \alpha, \alpha, \alpha, \alpha, \alpha, u_6, \dots, u_{15}, \alpha, \alpha, \alpha, \alpha, \alpha \rangle \\ \mathbf{W} &= \langle w_1, \dots, w_5, w_6, \dots, w_{15}, w_{16}, \dots, w_{20} \rangle . \end{aligned}$$

Let $\mathbf{M}'_i$ be a multiset similar to $\mathbf{M}_i$ except the first 2 elements of $\mathbf{X}$ are changed to be less than $\top$:

$$\begin{aligned} \mathbf{U}_{all} &= \langle u_1, \dots, u_{15} \rangle \\ \mathbf{X}' &= \langle x'_1, x'_2, x_3, x_4, x_5 \rangle \\ \mathbf{M}'_i &= \langle u_1, \dots, u_{13}, x'_1, x'_2, u_{14}, u_{15}, x_3, x_4, x_5 \rangle \\ \mathbf{V}'_i &= \langle \alpha, \alpha, \alpha, \alpha, \alpha, u_6, \dots, u_{13}, x'_1, x'_2, \alpha, \alpha, \alpha, \alpha, \alpha \rangle \\ \mathbf{W}' &= \langle w_1, \dots, w_5, w_6, \dots, w_{13}, w_{16}, w_{17}, w_{14}, w_{15}, \\ &\quad w_{18}, \dots, w_{20} \rangle . \end{aligned}$$

Then,

$$\begin{aligned} F(\mathbf{V}_i) &= \sum_{j=1}^{5} w_j \alpha + \sum_{k=6}^{15} w_k u_k + \sum_{\ell=16}^{20} w_\ell \alpha \\ F(\mathbf{V}'_i) &= \sum_{j=1}^{5} w_j \alpha + \sum_{k=6}^{13} w_k u_k + (w_{16} x'_1 + w_{17} x'_2) \\ &\quad + \sum_{m=14}^{15} w_m \alpha + \sum_{\ell=18}^{20} w_\ell \alpha. \end{aligned}$$

The diameter between the two weighted mean is:

$$\begin{aligned} F(\mathbf{V}_i) - F(\mathbf{V}'_i) &= \left(\sum_{k=6}^{15} w_k u_k - \sum_{k=6}^{13} w_k u_k \right) + \\ &\quad \left(\sum_{\ell=16}^{20} w_\ell \alpha - \sum_{\ell=18}^{20} w_\ell \alpha \right) - \\ &\quad \left(w_{16} x'_1 + w_{17} x'_2 + \sum_{m=14}^{15} w_m \alpha \right) . \end{aligned}$$

Therefore,

$$\begin{aligned} F(\mathbf{V}_i) - F(\mathbf{V}'_i) &= \sum_{k=14}^{15} w_k u_k + \sum_{\ell=16}^{17} w_\ell \alpha - \\ &\quad \left(w_{16} x'_1 + w_{17} x'_2 + \sum_{m=14}^{15} w_m \alpha \right) \\ &= \left(\sum_{k=14}^{15} w_k u_k - \sum_{m=14}^{15} w_m \alpha \right) + \\ &\quad \left(\sum_{\ell=16}^{17} w_\ell \alpha - (w_{16} x'_1 + w_{17} x'_2) \right) \\ &= \sum_{k=14}^{15} w_k (u_k - \alpha) + \\ &\quad \sum_{\ell=16}^{17} w_\ell (\alpha - x'_{\ell-15}). \end{aligned}$$

If $\alpha = \top$,

$$\begin{aligned} F(\mathbf{V}_i) - F(\mathbf{V}'_i) &= \sum_{k=14}^{15} w_k (u_k - \top) + \\ &\quad \sum_{\ell=16}^{17} w_\ell (\top - x'_{\ell-15}). \end{aligned}$$

Since $x'_{\ell-15} < \top$, the second term is positive. Also, the first term will be zero if u_k happens to be equal to $\top$. Given $\rho(\mathbf{U}_{all})$, this example showed that $F(\mathbf{V}_i)$ can produce a value larger than any other voted value if $\alpha = \top$ and no erroneous value is less than $\top$.

Lemma 1. *Let α be the value held by non-faulty node i and $\mathbf{V}_i$ be its voting multiset after replacing the extreme a elements of multiset $\mathbf{M}_i$. Assume the error multiset received by node i is $\mathbf{X} = \langle x_1, \dots, x_a \rangle$. $F(\mathbf{V}_i)$ can produce a voted value larger than any other one if $\alpha = \max(\mathbf{U}_{all})$ and $x_k \geq \max(\mathbf{U}_{all})$, $\forall\ k \in \{1, \dots, a\}$.*

Proof. According to the hypothesis:

$$\mathbf{M}_i = \langle u_1, \dots, u_{n-a}, x_1, \dots, x_a \rangle ,$$

so that

$$\begin{aligned} \mathbf{V}_i &= \langle \alpha, \dots, \alpha, u_{a+1}, \dots, u_{n-a}, \alpha, \dots, \alpha \rangle \\ \mathbf{W} &= \langle w_1, \dots, w_n \rangle \\ F(\mathbf{V}_i) &= \sum_{j=1}^{a} w_j\, \alpha + \sum_{k=a+1}^{n-a} w_k u_k + \sum_{\ell=1}^{a} w_{n-a+\ell}\, \alpha. \end{aligned}$$

Consider a different multiset $\mathbf{M}'_i$ with its corresponding $\mathbf{V}'_i$ and $\mathbf{X}'$. Assume $f < a$ elements of $\mathbf{X}'$ are less than $\max(\mathbf{U}_{all}) = \top$. This enables at most f elements of $\mathbf{U}_{all}$ to be pushed to the right, causing them to be replaced with α. Without loss of generality assume the first f elements of $\mathbf{X}'$ are less than $\max(\mathbf{U}_{all})$. Thus,

$$\begin{aligned} F(\mathbf{V}_i) - F(\mathbf{V}'_i) &= \sum_{k=n-a-f+1}^{n-a} w_k (u_k - \alpha) + \\ &\quad \sum_{\ell=n-a+1}^{n-a+f} w_\ell (\alpha - x'_{\ell-(n-a)}). \end{aligned}$$

Given $\rho(\mathbf{U}_{all})$, u_k can happen to be $\top$. If $\alpha = \top$,

$$F(\mathbf{V}_i) - F(\mathbf{V}'_i) = \sum_{\ell=n-a+1}^{n-a+f} w_\ell (\top - x'_{\ell-(n-a)}). \quad (3)$$

Since $x'_{\ell-(n-a)} \leq \top$, Equation (3) will be positive. □

Lemma 2. *Let β be the value held by non-faulty node j and $\mathbf{V}_j$ be its voting multiset after replacing the extreme a elements of multiset $\mathbf{M}_j$. Assume the error multiset received by node j is $\mathbf{X}_j = \langle x_1, \dots, x_a \rangle$. $F(\mathbf{V}_j)$ can produce a voted value smaller than any other one if $\beta = \min(\mathbf{U}_{all})$ and $x_k \leq \min(\mathbf{U}_{all})$, $\forall\ k \in \{1, \dots, a\}$.*

Proof. The proof is similar to Lemma 1. □

Let α and β be values of two arbitrary correct nodes, with the property of $|\alpha - \beta| \leq \varphi$, where φ is a predetermined positive threshold. Thus, $\delta(\mathbf{U}_{all}) = \varphi$. The rationale for φ is that in some applications, hosts, although working correctly, may not be able to produce the same precision, such as in sensor data management, clock synchronization, or in N-version programming applications [2, 15, 20].

Now we are ready to obtain $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ that will lead to determining C.

Theorem 1. *Given a WAA voting-algorithm,*

$$|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq \varphi(\text{sum of the highest } 3a \text{ weights}).$$

Proof. According to Lemma 1, $F(\mathbf{V}_i)$ can produce the largest voted value if $\alpha = \max(\mathbf{U}_{all})$ and no received error is less than $\max(\mathbf{U}_{all})$. Similarly, $F(\mathbf{V}_j)$ can reach its minimum value when $\beta = \min(\mathbf{U}_{all})$ and all erroneous values are less than or equal to $\min(\mathbf{U}_{all})$. Therefore, the maximum diameter of $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ is achieved when $\alpha - \beta = \varphi$. The values common to $\mathbf{V}_i$ and $\mathbf{V}_j$ are the correct values in $\mathbf{U}_{all}$ and thus can be considered fixed. The variable elements are associated with the faulty nodes, which can send conflicting values to nodes i and j. Without loss of generality, let the elements with weights $\langle w_{n-a+1}, \ldots, w_n\rangle$ be the variable elements shown as $\langle x_1, \ldots, x_a\rangle$ for node i and as $\langle x'_1, \ldots, x'_a\rangle$ for node j. Accordingly, node i receives the multiset:

$$\langle u_1, \ldots, u_a, u_{a+1}, \ldots, u_{n-a}, x_1, \ldots, x_a\rangle,$$

and node j has the multiset:

$$\langle x'_1, \ldots, x'_a, u_1, \ldots, u_{n-2a}, u_{n-2a+1}, \ldots, u_{n-a}\rangle.$$

Thus:

$$\mathbf{V}_i = \langle \alpha, \ldots, \alpha, u_{a+1}, \ldots, u_{n-a}, \alpha, \ldots, \alpha\rangle,$$

and

$$\mathbf{V}_j = \langle \beta, \ldots, \beta, u_1, \ldots, u_{n-2a}, \beta, \ldots, \beta\rangle.$$

Therefore:

$$\begin{aligned} F(\mathbf{V}_i) &= \sum_{g=1}^{a} w_g\alpha + \sum_{h=a+1}^{n-a} w_h u_h + \sum_{k=n-a+1}^{n} w_k\alpha \\ &= \sum_{g=1}^{a} w_g\alpha + \\ &\quad \left(\sum_{h=a+1}^{n-2a} w_h u_h + \sum_{m=n-2a+1}^{n-a} w_m u_m\right) + \\ &\quad \sum_{k=n-a+1}^{n} w_k\alpha, \end{aligned} \tag{4}$$

and

$$\begin{aligned} F(\mathbf{V}_j) &= \sum_{k=n-a+1}^{n} w_k\beta + \sum_{g=1}^{n-2a} w_g u_g + \sum_{m=n-2a+1}^{n-a} w_m\beta \\ &= \sum_{k=n-a+1}^{n} w_k\beta + \left(\sum_{g=1}^{a} w_g u_g + \sum_{h=a+1}^{n-2a} w_h u_h\right) + \\ &\quad \sum_{m=n-2a+1}^{n-a} w_m\beta. \end{aligned} \tag{5}$$

As a result:

$$\begin{aligned} |F(\mathbf{V}_i) - F(\mathbf{V}_j)| &= \sum_{g=1}^{a} w_g(\alpha - u_g) + \\ &\quad \sum_{m=n-2a+1}^{n-a} w_m(u_m - \beta) + \\ &\quad \sum_{k=n-a+1}^{n} w_k(\alpha - \beta). \end{aligned} \tag{6}$$

In Equation (6), maximum of $(\alpha - u_g)$ occurs if $u_g = \beta$. Similarly, $(u_m - \beta)$ is at its maximum if $u_m = \alpha$. Therefore:

$$\begin{aligned} |F(\mathbf{V}_i) - F(\mathbf{V}_j)| &\leq \sum_{g=1}^{a} w_g(\alpha - \beta) + \\ &\quad \sum_{m=n-2a+1}^{n-a} w_m(\alpha - \beta) + \\ &\quad \sum_{k=n-a+1}^{n} w_k(\alpha - \beta) \\ &= \sum_{g=1}^{a} w_g\varphi + \sum_{m=n-2a+1}^{n-a} w_m\varphi + \\ &\quad \sum_{k=n-a+1}^{n} w_k\varphi \\ &= \varphi\left(\sum_{g=1}^{a} w_g + \sum_{m=n-2a+1}^{n-a} w_m \right. \\ &\quad \left. + \sum_{k=n-a+1}^{n} w_k\right) \\ &= \varphi\left(\sum_{g=1}^{a} w_g + \sum_{m=n-2a+1}^{n} w_m\right). \end{aligned} \tag{7}$$

There are $3a$ weights in Equation (7). To maximize the difference in $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$, the weights in Equation (7) must be among the highest weights in $\mathbf{W}$. Therefore:

$$|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq \varphi(\text{sum of highest } 3a \text{ weights}). \tag{8}$$

□

Theorem 1 showed max$|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ when every element participates in the average function. If average function does not include every element, the sum of the selected weights no longer equals one. Therefore, the weighted averaging function can not be applied, unless those weights are normalized with respect to the weight sum of the selected elements, to ensure the sum of their weights is 1. Accordingly, function $F(\mathbf{V})$ as described in Equation (1) needs to be modified to account for the selection function, as explained below.

Let Sel_σ be the selection function that selects σ elements from $\mathbf{W}$:

$$\mathbf{S} = Sel_\sigma(\mathbf{W}) = \langle s_1, \ldots, s_\sigma \rangle .$$

Let g be the index of any element of $\mathbf{S}$ and let $k(g)$ be the index of the corresponding element in $\mathbf{W}$. Also reorder the elements in $\mathbf{V}_i$ such that $v_{i,g}$ corresponds to w_g. Therefore, for each $g \in \{1, \ldots, \sigma\}$ there exists exactly one $k(g) \in \{1, \ldots, n\}$. Define:

$$Sum(\mathbf{S}) = \sum_{g=1}^{\sigma} s_g,$$

so that the selected weights can be normalized as:

$$S^{nrm} = \langle s_1^{nrm}, \ldots, s_\sigma^{nrm} \rangle ,$$

where:

$$s_g^{nrm} = \frac{w_{k(g)}}{Sum(\mathbf{S})}.$$

Accordingly, each non-faculty node i applies the following approximation function:

$$F(\mathbf{V}_i) = \sum_{g=1}^{\sigma} s_g^{nrm} * v_{i,k(g)}.$$

The next lemma finds max$|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ when the selected weights are normalized, as described above.

Theorem 2. *Given a WAA voting-algorithm and a selection function Sel_σ,*

$$|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq \varphi(\text{sum of highest } 3a \text{ normalized weights in } Sel_\sigma).$$

Proof. According to WAA voting process, the nodes select the same weights. Resequence the weight indices of the selected elements symbolically, then normalize the weights. By assuming the a highest normalized weights belong to the faulty nodes, the process for finding max$|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ will become exactly the same as that of Theorem 1. □

Using Theorem 2 and the fact that $\delta(\mathbf{U}_{all}) = \varphi$, the convergence rate is:

$$\begin{aligned} C &= \frac{\delta(\mathbf{U}_{voted})}{\delta(\mathbf{U}_{all})} \\ &= \frac{\max|F(\mathbf{V}_i) - F(\mathbf{V}_j)|}{\delta(\mathbf{U}_{all})} \\ &= \frac{\varphi(\text{sum of highest } 3a \text{ normalized weights in } Sel_\sigma)}{\varphi} \\ &= (\text{sum of highest } 3a \text{ normalized weights in } Sel_\sigma). \quad (9) \end{aligned}$$

3.4 Fault-Tolerance

Recall that the sum of weights equals 1 and convergence is guaranteed if $C < 1$. Therefore, more than $3a$ elements must be selected to ensure Equation (9) is less than 1, which implies $|\mathbf{V}| > 3a$. Consequently, to tolerate a faults, $|\mathbf{V}| = n \geq 3a + 1$. This limit can also be observed in Equations (4) and (5) of Theorem 1. More specifically, the term:

$$\sum_{h=a+1}^{n-2a} w_h u_h, \quad (10)$$

is cancelled in $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$. If Equation (10) does not exist, the sum of weights in Equation (8) will equal one. For this term to exist $n - 2a \geq a + 1$ must be true, which implies $n \geq 3a + 1$.

Since $n = N - b$, the total number of hosts to guarantee the existence of a convergent WAA algorithm must be:

$$N \geq 3a + b + 1.$$

3.5 Working Example 2

Consider a system with 14 nodes, i.e. $n = 14$, with the following normalized weights:

$w_1 = 0.01$	$w_2 = 0.02$	$w_3 = 0.03$
$w_4 = 0.03$	$w_5 = 0.05$	$w_6 = 0.05$
$w_7 = 0.06$	$w_8 = 0.07$	$w_9 = 0.08$
$w_{10} = 0.1$	$w_{11} = 0.1$	$w_{12} = 0.11$
$w_{13} = 0.14$	$w_{14} = 0.15$	

Assume $a = 3$ and $\sigma = 12$. Further assume that the selection function selects the lowest weights in $\mathbf{W}$. Under these assumptions, the sum of the selected weights is 0.71. After normalization over the 12 weights, the new weights will be:

$$0.014, 0.028, 0.042, 0.042, 0.071, 0.071,$$
$$0.085, 0.099, 0.113, 0.140, 0.141, 0.154.$$

For example, the lowest weight under normalization is $0.01/0.71 = 0.014$. The convergence rate is the sum of the highest $3a = 9$ normalized weights , i.e. $C = 0.916$.

Now consider a different example with the same assumptions except the selection function selects the highest weights. With this change, the sum of the 12 weights is 0.97 and the new normalized weights are:

$$0.031, 0.031, 0.052, 0.052, 0.061, 0.072,$$
$$0.082, 0.104, 0.104, 0.113, 0.144, 0.154.$$

Under this new example, $C = 0.886$. Therefore, the choice in selecting weights affects the convergence rate. Consequently, depending on application, an appropriate distribution of weights and selection function must be applied to reach the desired level of performance. For instance, in a harsh environment, one may wish to distribute low weights to a large number of nodes to ensure a better average. On the other, if some nodes can be better trusted than others, one can use higher weights for these nodes, to tilt the final results toward the values held by such nodes.

4 The Optimal Convergence Rate

As shown in Theorem 2, the convergence rate is the sum of the largest $3a$ normalized weights. The optimal convergence occurs when C is the lowest possible. Therefore, out of all selections of σ elements, one needs to find the selection function such that the sum of its highest $3a$ elements is the lowest among all selection functions. The following shows how to obtain the optimal convergence rate formally, for a given multiset of weights.

The total number of possible selections of σ elements out of n elements in $\mathbf{W}$ is:

$$C_{n,\sigma} = \binom{n}{\sigma} = \frac{n!}{\sigma!(n-\sigma)!}. \qquad (11)$$

After normalization, sort each selection and store in $\mathbf{S}_k^{nrm}$:

$$\mathbf{S}_k^{nrm} = \left\langle s_{k,1}^{nrm}, \ldots, s_{k,\sigma}^{nrm} \right\rangle, \text{where } k \in \{1, \ldots, C_{n,\sigma}\}.$$

Partition $\mathbf{S}_k^{nrm}$ into two sub-multisets as:

$$First_k = \sum_{i=1}^{\sigma-3a} s_{k,i}^{nrm}$$
$$Last_k = \sum_{i=\sigma-3a+1}^{\sigma} s_{k,i}^{nrm}.$$

The optimal convergence rate occurs when a selection function, $\mathbf{S}_q^{nrm}$, is chosen with the following property:

$$(Last_q - First_q) \leq (Last_j - First_j), \ \forall \, j \in \{1, \ldots, C_{n,\sigma}\}.$$

Looking back at the example used before in Section 3.5, the highest 12 weights provide the best convergence, with $Last = 0.886$, $First = 0.114$, and $Last - First = 0.772$. The worst convergence occurs when w_3 and w_5 are not selected with $First = 0.065$, $Last = 0.935$, $Last - First = 0.870$, and $C = 0.935$.

5 Performance Adjustment of WAA Algorithms

WAA algorithms have the capability of determining the feasibility of a convergence rate with regard to the upper bound of faulty nodes. Assume the minimum weight value in the highest $3a$ normalized weights is m. Given the desired C, the following relationship shows the upper bound on the faulty nodes allowed:

$$\frac{C}{m} \geq 3a, \quad where\ C > m. \qquad (12)$$

For instance if a convergence rate of $C = 0.5$ is desired and the minimum weight among the $3a$ highest normalized weights is 0.1, then only one fault, i.e. $a = 1$, can be tolerated.

Furthermore, $(1 - C)$ provides the sum of the weights for the rest of the selected elements. Since the weight of each such element can not be greater than m, the least number of such nodes is:

$$\frac{1-C}{m}.$$

As a result, n must satisfy the following, given that Equation (12) is true:

$$n \geq \frac{1-C}{m} + 3a.$$

Using the previous example of $m = 0.1$, requiring $C = 0.5$ and $a = 1$, at least 8 nodes, i.e. $n = 8$, are needed.

Recall from Section 3.4 that $n \geq 3a + 1$. If $a = 1$, the minimum number of nodes needed is: $n \geq 3a + 1 = 4$, which appears to contradict with the minimum value of 8 obtained in this section. However, the reader should note that the expression $n \geq 3a + 1$ shows the absolute minimum value to ensure the existence of a convergent voting algorithm without any regards to the rate of convergence that one might desire for.

Requiring both conditions of Agreement and Validity of WAA algorithms guarantees *precision* and *accuracy*. Precision refers to the tightness of decision values among the nodes, and thus corresponds to the Agreement condition. Accuracy, on the other hand, defines the deviation of decision values from the real correct result, and thus corresponds to the Validity condition. The precision criterion alone might be justified for some applications, such as synchronization of internal clocks for the purpose of coordinating events among processes.

WAA algorithms replace the lowest and the highest a values by the node's own value to ensure the Validity condition. The replacement however creates more discrepancies among nodes. By relaxing this condition, it might be possible to find a way to increase the performance of WAA algorithms. This approach is currently under investigation.

6 Comparison to other Algorithms

Other than WAA, other families of AA algorithms have been developed, such as Mean-Subsequence-Reduced (MSR) [12], Mean-Subsequence-Egocentric (MSE) [5], and Mean-Subsequence-Egophobic (MSEP) [3]. These algorithms are different than WAA in the following:

- Selection of elements in MSR, MSE, and MSEP is done without any attention to the trustworthiness of the hosts. A selection function selects values from the voting multiset based on their positions. For example, a selection function may select all odd-numbered positions. The selection process used for these algorithms is thus oblivious. This has the impact that the set of hosts corresponding to the selected values for a pair of hosts may not be the same. Whereas in WAA, the selection function is based on $\mathbf{W}$. Thus the selected hosts, whose received values are used in the voting process, are the same among all nodes.
- In other voting algorithms, since the hosts are considered to have the same trust-weight, each host carries the weight of $1/n$. Whereas in WAA, nodes can have different weights.
- Unlike other voting algorithms, a system designer by using higher weights for trustworthy hosts can tilt the decision values toward the values held by those hosts.

Because the selection function of WAA algorithms is based on weights, it may not be possible to apply the same selection function to all families of algorithms. For example, the Fault-Tolerant Midpoint [12], which averages the extreme two elements of the voting multiset, is not applicable to WAA.

One selection function that is immune to these differences is the Fault-Tolerant Mean [12]. In Fault-Tolerant Mean, every element in $\mathbf{V}$ is selected. By equalizing the weights in WAA, i.e. $1/n$, this selection function can be applied to all families of algorithms mentioned here.

Ensuring the conditions of agreement and validity, the convergence rates in MSR, MSE, and MSEP using the Fault-Tolerant Mean are the same [3, 5, 12]. The convergence rate is:

$$C = \frac{a}{n-2a}.$$

Since the trust level is assumed to be the same for each host, the convergence rate in WAA is:

$$C = \frac{3a}{n}.$$

The convergence rate in WAA using this selection function is thus lower than that of MSR, MSE, and MSEP. It should be observed, however, that the major advantage of WAA algorithms is the ability of using varying weights.

Using the hybrid fault modeling, such as incorporating the symmetric faults, used in MSR or MSE does not affect the performance or the fault tolerance of the WAA algorithms. A symmetric fault occurs when a faulty host transmits the same erroneous value to the receiving hosts. Adding the symmetric faults to the fault model forces hosts to replace more of the extreme values with their own to ensure that the validity condition is met. Since hosts hold different values, the impact of other failure modes, such as symmetric faults, thus becomes like asymmetric faults.

7 Summary

A number of studies have been done in AA with the goal of creating families of algorithms whose convergence rate and fault tolerance can be easily determined. Some of these families were mentioned in this paper. These algorithms treat every host the same, i.e. the trust-weight placed on them are all equal. It is very difficult to apply the weighted voting concept to these algorithms because their premise in determining the convergence rate is based on comparing values selected from the sorted voting multisets. If varying weights are used, the weight multisets corresponding to the voting multisets may not all be the same, which makes it difficult to keep track of the relationships among terms produced by the weighted voting function $F(\mathbf{V})$. This difficulty is compounded when the weights of the selected elements are normalized, which completely destroys any cohesiveness that might exist among selected elements at different hosts.

Because of these difficulties, this study created a new family of algorithms called WAA, with the property that the selection is applied to the weights rather than on the sorted values held by hosts. This ensures that hosts will use the same weights in a round of voting process. As this is the first study on weighted voting for approximate agreement algorithms, it is not possible to compare the WAA algorithms against other families that use weights to reach consensus in the presence of faults.

As a voting algorithm has a better performance when the convergence rate C is lowered, and the fact that the convergence rate of WAA algorithms depends on the highest $3a$ normalized weights, it becomes important to adjust the weights so that the performance is improved, yet the decision values are leaned toward the values held by hosts with higher confidence level. In general, WAA algorithms show good performance when the number of hosts is relatively larger than $3a$.

With the insight obtained from this study, it will be worthwhile to consider other strategies in reaching agreement for weighted AA algorithms, such as using maximal sets with the addition of incorporating different failure modes. Another avenue of study is to challenge the feasibility of applying weighted voting to existing algorithms such as MSR.

References

[1] M. Ahmmad, and M.H. Ammar, "Performance characterization of quorum-consensus algorithms for replicated data," *Proceedings of the 7th Symposium on Reliability in Distributed Software and Database Systems*, pp. 161-167, 1987.

[2] A. Avizienis, "The n-version approach to fault-tolerant Software," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1491-1501, Dec. 1985.

[3] M. H. Azadmanesh, and L. Zhou, "Egophobic voting algorithms," *Journal of Computers & Applications*, vol. 25, no. 4, pp. 236-246, 2003.

[4] M. H. Azadmanesh, and R.M. Kieckhafer, "Exploiting omissive faults in synchronous approximate agreement," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1031-1042, Oct. 2000.

[5] M. H. Azadmanesh, and A. W. Krings, "Egocentric voting algorithms," *IEEE Transactions on Reliability*, vol. 46, no. 4, pp. 494-502, Dec. 1997.

[6] M. Barborak, M. Malek, and A. Dahbura, "The consensus problem in fault tolerant computing," *ACM Computing Surveys*, pp. 171-220, Jun. 1993.

[7] R. Brooks, and S. Iyengar, "Robust distributed computing and sensing algorithm," *IEEE Computer*, pp. 53-60, June 1996.

[8] T. Clouqueur, K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 320-333, Mar. 2004.

[9] D. Dolev, et al., "Reaching approximate agreement in the presence of faults," *Journal of ACM*, vol. 33, no. 3, pp. 499-516, Jul. 1986.

[10] D. K. Gifford, "Weighted voting for replicated data," *Proceedings of the 7th Symposium on Operating Systems Principles*, ACM SIGOPS, pp. 150-159, 1979.

[11] J. L. W. Kessels, "Two designs of a fault-tolerant clocking system," *IEEE Transactions on Computers*, vol. C-33, no. 10, pp. 912-919, Oct. 1984.

[12] R. M. Kieckhafer, and M. H. Azadmanesh, "Reaching approximate agreement with mixed mode faults," *IEEE Transactions on Parallel Distributed Systems*, vol. 5, no. 1, pp. 53-63, Jan. 1994.

[13] C. M. Krishna, K. G. Shin, and R. W. Butler, "Ensuring fault tolerance of phase-locked clocks", *IEEE Transactions on Computers*, vol. C-34, no. 8, pp. 752-756, Aug. 1985.

[14] C. M. Krishna, "Fault-tolerant synchronization using phase-locked clocks", *Microelectronics Reliability*, vol. 30, no. 2, pp. 275-287, 1990.

[15] L. Lamport, P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults", *Journal of ACM*, vol. 32, no. 1, pp. 52-78, Jan. 1985.

[16] K. Marzullo, "Tolerating failures of continuous-valued sensors", *ACM Transactions on Computer Systems*, vol. 8, no. 4, pp. 284-304, Nov. 1990.

[17] B. Parhami, "Voting algorithms", *IEEE Transactions on Reliability*, vol. 43, no. 4, pp. 617-629, Dec. 1994.

[18] P. Ramanathan, D. D. Kandlur, and K. G. Shin, "Hardware-assisted software clock synchronization for homogeneous sistributed systems", *IEEE Transactions on Computers*, vol. C-39, no. 4, pp. 514-524, Apr. 1990.

[19] P. Ramanathan, K. G. Shin, and R. W. Butler, "Fault-tolerant clock synchronization in distributed systems," *IEEE Computer*, 23, pp. 33-42, Oct. 1990.

[20] P. M. Thambidurai, et al., "Clock Synchronization in MAFT," *Proceedings of Nineteenth Fault-Tolerant Computing Symposium*, pp. 142-151, June 1989.

[21] Z. Tong, and R.Y. Kain, "Vote Assignments in Weighted Voting Mechanism", *Proceedings of the 7th Symposium on Reliable Distributed Systems*, pp. 138-143, 1988.

[22] J. Wu, and R. Chen, "Efficient Algorithms for k-out-of-n & Consecutive-weighted-k-out-of-n system", *IEEE Transactions on Reliability*, vol. 43, no. 4, pp. 327-328, June 1994.

Azad Azadmanesh received the BS degree in Cost Accounting. He received his MS and PhD degrees in Computer Science from Iowa State University and University of Nebraska-Lincoln, USA, respectively. He is currently a professor in the Department of Computer Science at University of Nebraska-Omaha. His research interests include Security and Survivability of Network Systems, Fault-Tolerance, Reliability Modeling, and Distributed Agreement.

Alireza Farahani received his BS degree in Mathematics from state University of New York at Buffalo. He completed his MS and PhD degrees in Computer Science and Applied Mathematical Sciences respectively from University of Rhode Island. He is currently an associate professor in the Department of Computer Science and Information Systems at National University in San Diego, California. His research interests include Optimization, Control Theory and Algorithm Design.

Lotfi Najjar is an Assistant Professor in the Department of Information Systems and Quantitative Analysis (ISQA) in the College of Information Science and Technology at University of Nebraska-Omaha. He holds a PhD in Industrial and Management Systems Engineering with supporting areas in MIS and Operations Management from University of Nebraska-Lincoln. His research interests are in the areas of Quality Information Systems in the service (banking industries) and manufacturing industries, Systems Reliability and Quality, Business Process Reengineering, Data Mining, and Total Quality Management (TQM) & IT.