Asynchronous Behavior of Egoestic Voting Algorithms*

M. H. Azadmanesh
Computer Science Department
University of Nebraska at Omaha
Omaha, NE 68182-0500, USA
azad@ahvaz.unomaha.edu

and

A. W. Krings
Computer Science Department
University of Idaho
Moscow, ID 83844-1010, USA
krings@cs.uidaho.edu

Abstract

The Approximate Agreement problem is a variant of the Byzantine General Problem in which processes start with real values, rather than with boolean values, and each process agrees on an approximate value close to the true real value.

Recent studies showed the performance of Egocentric and Egophobic algorithms under the synchronous model of communication. However, the Egocentric and Egophobic algorithms have never been analyzed for the asynchronous model under hybrid fault models. Therefore this paper shows the fault tolerance and performance of Egocentric and Egophobic algorithms under the asynchronous model, and shows how different these algorithms are in comparison to the algorithms in the synchronous model. The completion of this study is a spring-board for identifying a unified model representing the synchronous and asynchronous communication models of Egocentric and Egophobic algorithms. This unified model can most likely be extrapolated to include other families of voting algorithms.

Keywords – Approximate Agreement, Byzantine Agreement, Clock Synchronization, Convergent Voting Algorithms, Hybrid Fault Models.

*This work was supported by the College of Information Science & Technology (IS&T) Research Program.

1 Introduction

The Approximate Agreement Problem (AAP) [6, 7] is a form of agreement in which the processes start with real values and need to reach an approximate agreement rather than an exact agreement. AAP can be used, for example, in synchronization of distributed clocks, or for applications where the stabilization of sensors is required.

The objective of reaching Approximate Agreement is to guarantee that, at the termination of a voting process, the voted value for each non-faulty process is within the range of the initial correct values and that the difference between any pair of voted values for non-faulty processes is strictly within a prespecified small positive real value ϵ. The final voted value, at the end of the voting algorithm, is obtained by employing multiple rounds of message exchange. In each round, each process sends its value to all receiving processes. Upon receipt of a collection of values, each process executes a function F, to obtain its latest voted value, which is used in the next round of message exchange. This objective is achieved by ensuring that each round is convergent, i.e. the range of the correct values is reduced in each round [6, 7]. This property, called *single-step convergence*, guarantees that the range of values will eventually be less than ϵ, given enough rounds.

Two categories of voting algorithms have been studied recently: *Egocentric* [4], and *Egophobic* [3]. In Egocentric algorithms, the processes place more trust on their own values than the other received values. Consequently, if an error is detected or a value is not received, a process tends to correct the error by replacing it with its own value. For Egophobic algorithms, a process places more trust on other values than its own, thus replacing the erroneous values with a value other than its own.

The communication model can be *synchronous* or *asynchronous*. In a synchronous communication system, the processing and communication delays of non-faulty processes are bounded. Any data arriving after this time is considered erroneous. In contrast, an asynchronous system imposes no finite bounds on process operations or communication delays.

Egocentric and Egophobic, referred to, in general, as Egoestic algorithms, have been studied under the synchronous model, but not under the asynchronous model of communication. Therefore this paper analyzes the performance and fault tolerance of Egoestic algorithms under the asynchronous model of communication, and compares the results with those of their counter-part synchronous Egoestic and other asynchronous voting algorithms.

The rest of the paper is organized as follows. Section 2 defines the system model and some definitions. Section 3 introduces the Egoestic algorithms, and gives the convergence rate for their synchronous model of communication. It also obtains their convergence rate and fault tolerance for the asynchronous form of communication. Section 4 provides some examples, showing how to use the results of Section 3 to obtain the convergence rate. Section 5 summarizes the paper, and provides some remarks for further study of voting algorithms.

2 System Model & Definitions

The system model is a completely connected network consisting of N nodes (processes). The timing of communication is *asynchronous*. We use a hybrid fault model consisting of *benign*, *symmetric*, and *asymmetric* faults [7]. Benign faults are self-incriminating or self-evident to all processes. A symmetric fault is defined as a fault whose erroneous value is perceived identically by all receiving non-faulty processes, whereas an asymmetrically faulty process sends conflicting messages to non-faulty processes.

In an asynchronous system, a non-faulty process may never receive values from all processes in the system. Thus, the process only waits to receive the first $n = N - t$ values before voting on a new value, where t is the maximum number of malicious faulty processes in the system. Furthermore, no fault may be considered benign, since any value transmitted may not be included by all processes in their voting multisets. Thus, an error is not immediately evident to all non-faulty processes. Hence, given a maximum of a asymmetric and s symmetric faults, the total number of faults is $t = a + s$.

Single-Step Convergence – AAP requires the manipulation of multisets of real values. A multiset of real numbers can be represented as a monotonically increasing sequence of the real values of its elements, i.e. $\mathbf{V} = \langle v_1, \ldots, v_V \rangle$ is ordered such that: $v_i \leq v_{i+1} \; \forall \; i \in \{1, \ldots, V-1\}$. The size of $\mathbf{V}$ is $V = |\mathbf{V}|$. Among many multiset operations, two of the operations are:

- $\rho(\mathbf{V}) = [\min(\mathbf{V}), \max(\mathbf{V})]$, which is the real interval spanned by $\mathbf{V}$, and

- $\delta(\mathbf{V}) = \max(\mathbf{V}) - \min(\mathbf{V})$, which is the diameter of $\mathbf{V}$.

During each round of voting, each non-faulty process i broadcasts its initial value to all processes including itself. Then it collects the first n values into the *voting multiset* $\mathbf{V}_i$. Process i then applies a function F to $\mathbf{V}_i$ to attain its latest estimate (voted value) for the round, which is used as the value of the process to be broadcast in the next round.

To formally define the property of single-step-convergence, define $\mathbf{U}_{all}$ as the multiset of all correct values received by non-faulty processes. An algorithm is single-step-convergent if both of the following conditions are true following every round of voting:

- For each non-faulty process i, the voted value is within the range of correct values, i.e. $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{all})$

- For each pair of non-faulty processes, i and j, the difference between their voted values is *strictly less than* the diameter of the multiset of correct values received, i.e. $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| < \delta(\mathbf{U}_{all})$.

Assuming that $\delta(\mathbf{U}_{all}) > 0$, the maximum of the ratio:

$$C = |F(\mathbf{V}_i) - F(\mathbf{V}_j)| / \delta(\mathbf{U}_{all})$$

shows the effectiveness (convergence rate) of a convergent voting algorithm. If $C < 1$ in each round, it is then guaranteed that after enough number of rounds, the system will achieve agreement, i.e. the diameter of correct values held by the non-faulty processes in the system will be within ϵ.

3 Egoestic Voting Algorithms

Define α and β as the values of two arbitrary non-faulty processes i and j in a particular round respectively. Also, define $v_{i,h}$ and $v_{j,l}$ as two values in the voting multisets $\mathbf{V}_i$ and $\mathbf{V}_j$ respectively. Egoestic algorithms then have the following properties:

$$|\alpha - \beta| \leq \varphi, \ \ |v_{i,h} - \alpha| \leq \varphi, \ \text{and} \ \ |v_{j,l} - \beta| \leq \varphi$$

where φ is a positive real value. The rationale for φ is that in some applications processes may not all produce exactly the same value. The discrepancy among processes might be, for instance, due to clocks which must stay within a predefined known bound of each other, or multiple software or hardware versions where decision-algorithms are employed to determine the final vote from similar but not identical results [1, 8].

Two families of Egoestic algorithms have been studied, called Mean-Subsequence-Egophobic (MSEP) and Mean-Subsequence-Egocentric (MSE) [3, 4]. These algorithms use the voting function:

$$F(\mathbf{V}_i) = \text{mean}\left[Sel_\sigma\left(\mathbf{V}_i\right)\right]$$

Sel_σ selects a submultiset of σ elements from the voting multiset $\mathbf{V}_i$. $F(\mathbf{V}_i)$ is the mean of these selected elements. A process considers a received value erroneous if it is different from its own value by more than φ. In MSE, a process i holding the value α replaces the erroneous value with its own value α. A process i in MSEP, however, replaces the erroneous value with the farthest value from its own, i.e., $(\alpha + \varphi)$.

An important parameter to Egoestic algorithms is γ. The purpose of γ is to show whether a voting algorithm is convergent. If so, it will determine the convergence rate. The merits of γ will become clear when the convergence rate expression is defined in the next subsection.

Definition of γ – The selected multiset $\mathbf{S} = Sel_\sigma(\mathbf{V}) = \langle s_1, \ldots, s_\sigma \rangle$ is a *subsequence* of $\mathbf{V} = \langle v_1, \ldots, v_n \rangle$. Thus, each element of $\mathbf{S}$ corresponds to one unique element of $\mathbf{V}$. Let g be the index of any element of $\mathbf{S}$, and let $k(g)$ be the index of the corresponding element in $\mathbf{V}$, so that $s_g = v_{k(g)}$. Now given two indices into $\mathbf{S}$, g and $h \in \{1, \ldots, \sigma\}$, where $g \leq h$, define $\Delta k(g,h) = k(h) - k(g)$ as the number of elements in $\mathbf{V}$ *spanned* by $(s_g, \ldots, s_h)$ in $\mathbf{S}$.

Finally, for any non-negative integer z, define γ_z as the minimum value of $(h - g)$ which ensures that $k(h) - k(g) \geq z$, for any values g and h, $g \leq h \leq \sigma$. By this definition, γ_z exists only if $|\mathbf{V}| > z$.

3.1 Convergence Rate of Synchronous Egoestic Voting Algorithms

By the definition of convergence rate, to obtain C, the maximum of $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$ must be determined. Ultimately, one needs to find the sum-difference of the selected elements in $\mathbf{S}_i$ and $\mathbf{S}_j$, because $F(\mathbf{V})$ is defined in terms of the selection function $Sel_\sigma(\mathbf{V})$.

Let ω_{γ_z} be a weight function that uses two lists that show the weight of each element in $\mathbf{V}_i$ and $\mathbf{V}_j$. The maximum difference in these weights will determine the value of ω_{γ_z}. These lists are called *Enumerated Selected Lists* $E_i\_Sel = \langle e_{i,1}, \ldots, e_{i,\sigma} \rangle$ and $E_j\_Sel = \langle e_{j,1}, \ldots, e_{j,\sigma} \rangle$, and are associated with processes i and j respectively. The g^{th} element of E_i_Sel represents the weight of $v_{i,k(g)}$ in $\mathbf{V}_i$ if it is in $Sel_\sigma(\mathbf{V}_i)$. E_j_Sel is defined similarly.

For synchronous MSE [4], it has been shown that $e_{i,x}$ and $e_{j,y}$ have the following weights:

$$e_{i,x} \ = \ \begin{cases} 1 & : \ k(x) = 1 \\ 2 & : \ 2 \leq k(x) \leq (N - a - s), \\ 3 & : \ \mathit{otherwise} \end{cases}$$

and

$$e_{j,y} \ = \ \begin{cases} 0 & : \ 1 \leq k(y) \leq a \\ 1 & : \ \mathit{otherwise} \end{cases}$$

For synchronous MSEP algorithms [3], it has been shown that:

$$e_{i,x} \ = \ \begin{cases} 1 & : \ k(x) = 1 \\ 2 & : \ 2 \leq k(x) \leq (N - a - s), \\ 3 & : \ \mathit{otherwise} \end{cases}$$

and

$$e_{j,y} \ = \ \begin{cases} 0 & : \ 1 \leq k(y) \leq (a + s) \\ 1 & : \ \mathit{otherwise} \end{cases}$$

Now define the weight function ω_{γ_z} to be:

$$\omega_{\gamma_z} = \sum_{g=1}^{\gamma_z} [e_{i,\sigma-g+1} - e_{j,g}] \tag{3.1}$$

Finally, it has been shown [3, 4] that the convergence rate is:

$$C = \frac{\omega_{\gamma_{(a+s)}}}{\sigma} \tag{3.2}$$

3.2 Convergence Rate of Asynchronous Egoestic Voting Algorithms

THEOREM 1 : *Given an Asynchronous Egoestic voting-algorithm* $F(\mathbf{V})$,

$$C = \frac{\omega_{\gamma_{2t}}}{\sigma} \tag{3.3}$$

PROOF: Consider two arbitrary non-faulty processes i and j for a particular round of voting. The processes i and j must wait to receive the first n values before producing their voted values for the round. Since there are $N = n + t$ processes, in the worst case process i can receive values from t non-faulty processes that are not received by process j. In this worst case, process j must have received values from a different set of t non-faulty processes whose values are not received by process i. Since there are t pairs of correct values, each pair can mimic the behavior of an asymmetric fault.

Still in the worst case, both processes can receive values from t faulty processes. In addition, similar to the synchronous case, Theorems 2 & 3 have shown that in the worst case, one of the processes, i or j, must replace the value of a symmetric fault with a default value, transforming the fault into an asymmetric fault. As a result, an asynchronous system with t faulty processes behaves essentially like a synchronous system with 2t faulty processes. Since $t = a + s$, by replacing t with $2t$ in (3.2), (3.3) is obtained. □

The convergence rate expression for both MSE and MSEP algorithms is the same. However, they do not produce the same rate of convergence because the weights in the enumerated selected lists are different. The following theorems show how these weights are obtained.

THEOREM 2 : *Given* $E_i\_Sel_\sigma$ *and* $E_j\_Sel_\sigma$, *and the arbitrary non-faulty processes* i *and* j, *the weights for Asynchronous MSE voting algorithms are:*

$$e_{i,x} = \begin{cases} 1 & : \ k(x) = 1 \\ 2 & : \ 2 \leq k(x) \leq (n-a-s), \\ 3 & : \ otherwise \end{cases} \tag{3.4}$$

and

$$e_{j,y} = \begin{cases} 0 & : \ 1 \leq k(y) \leq a \\ 1 & : \ otherwise \end{cases} \tag{3.5}$$

PROOF: Consider the two non-faulty processes i and j holding the values α and β. Since we need to know the maximum of $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$, $F(\mathbf{V}_i)$ and $F(\mathbf{V}_j)$ must be maximized and minimized respectively. This implies that $\alpha - \beta$ must be the maximum possible, i.e. $|\alpha - \beta| = \varphi$. Thus, without loss of generality, assume $\beta = \alpha - \varphi$.

To maximize $|F(\mathbf{V}_i) - F(\mathbf{V}_j)|$, it is shown [4] that asymmetric and symmetric faulty processes must transmit their maximum allowable values without allowing for process i to replace them with its own value α. Thus, these processes must transmit $(\alpha + \varphi)$ to process i.

To minimize $F(\mathbf{V}_j)$, the asymmetric faulty processes will transmit the smallest allowable values to process j, i.e. $\beta - \varphi = \alpha - 2\varphi$. Since the symmetric faulty processes must transmit the same values to both i and j, and they transmitted $(\alpha + \varphi)$ to process i, their values will be out of bound for process j, since $(\alpha + \varphi) \geq (\beta + \varphi)$. Thus, process j replaces these values with its own value $\beta = \alpha - \varphi$.

Theorem 1 showed that an asynchronous system with t faulty processes behaves like a synchronous system with $2t$ faults. However, the t pairs of values that are received by one process, such as i, and not received by the other process, such as j, are correct values, and thus must have the maximum diameter of φ. In addition, $\mathbf{V}_i$ contains both the correct values for i and j, i.e. α and $\alpha - \varphi$ respectively, and $F(\mathbf{V}_i)$ and $F(\mathbf{V}_j)$ must be maximized and minimized respectively. Therefore process i must receive the highest possible value of α from t non-faulty processes, and process j must receive the smallest values of $\alpha - \varphi$ from a different t non-faulty processes. Considering all the conditions just described forms the following in the worst case:

$$\begin{aligned}\mathbf{V}_i &= \langle \alpha-\varphi, \alpha, \ldots, \alpha, \alpha+\varphi, \ldots, \alpha+\varphi\rangle & (3.6)\\ \mathbf{V}_j &= \langle \alpha-2\varphi, \ldots, \alpha-2\varphi, \alpha-\varphi, \ldots, \alpha-\varphi\rangle & (3.7)\end{aligned}$$

In (3.6), the first $n-2t$ elements, i.e. $\langle \alpha-\varphi, \alpha, \ldots, \alpha\rangle$, belong to the correct values. The first element $(\alpha-\varphi)$ of $\mathbf{V}_i$ is the value sent by process j, and the second element α is the value of process i. The next t elements in (3.6) having the value α are also correct coming from t correct processes whose values are not received by process j. The rest of the elements, i.e. $\langle \alpha+\varphi, \ldots, \alpha+\varphi\rangle$, are the asymmetric and symmetric errors holding their maximum values. Similarly, the first a elements of (3.7) having the value $(\alpha-2\varphi)$ belong to the asymmetric faulty processes. The next s elements with value $(\alpha-\varphi)$ are those symmetric values which were transmitted as $(\alpha+\varphi)$ and replaced by $\beta=(\alpha-\varphi)$. The next t values in the $(\alpha-\varphi)$ series come from the t non-faulty processes which have not been received by process i. The rest are the correct values which are common between processes i and j. Since any pair of correct values $v_{i,k}$ and $v_{j,l}$ in $\mathbf{V}_i$ and $\mathbf{V}_j$ must have the property of: $|v_{i,k}-v_{j,l}| \le \varphi$, we have purposely used α in $\mathbf{V}_i$ and $\alpha-\varphi$ in $\mathbf{V}_j$ to ensure that, regardless of the way in which the elements are selected, $|s_{i,g}-s_{j,g}|$ is maximized.

In (3.6) and (3.7), let the elements with values $(\alpha-2\varphi)$, $(\alpha-\varphi)$, α, and $(\alpha+\varphi)$ carry the weights $0, \ldots, 3$, respectively. Using these weights, $\mathbf{V}_i$ and $\mathbf{V}_j$ can be converted into the following enumerated lists:

$$\begin{aligned}E_i\_Sel_\sigma &= \langle 1, 2, \ldots, 2, 3, \ldots, 3\rangle & (3.8)\\ E_j\_Sel_\sigma &= \langle 0, \ldots, 0, 1, \ldots, 1\rangle & (3.9)\end{aligned}$$

In (3.8), there are $t=a+s$ elements with weight 3, and there are $(n-t-1)$ elements with weight 2. Similarly, in (3.9), there are a elements with weight 0, and $(n-a)$ elements with weight 1. The weights in (3.4) are merely a different representation of (3.8) and (3.9) respectively. □

THEOREM 3 : *Given $E_i\_Sel_\sigma$ and $E_j\_Sel_\sigma$, and the arbitrary non-faulty processes i and j, the weights for Asynchronous MSEP voting algorithms are:*

$$e_{i,x} = \begin{cases} 1 & : k(x)=1\\ 2 & : 2\le k(x)\le (n-a-s),\\ 3 & : otherwise\end{cases} \quad (3.10)$$

and

$$e_{j,y} = \begin{cases} 0 & : 1\le k(y)\le (a+s)\\ 1 & : otherwise\end{cases} \quad (3.11)$$

PROOF: The discussion on the t pairs of correct values and the method to maximize $|F(\mathbf{V}_i)-F(\mathbf{V}_j)|$ is the same as that for asynchronous MSE voting algorithms. Also as shown in [3], and indicated in Theorem 2, process j must have the value $\beta=\alpha-\varphi$ to ensure that $|F(\mathbf{V}_i)-F(\mathbf{V}_j)|$ is maximized.

In regard to the t faulty processes, according to synchronous MSEP voting algorithms [3], the symmetric and asymmetric faulty processes must transmit the smallest values, i.e. $\alpha-2\varphi$ to both processes i and j to guarantee that $|F(\mathbf{V}_i)-F(\mathbf{V}_j)|$ is maximized. Process i replaces these out of bound values by the furthest allowable value from its own, i.e. $\alpha+\varphi$. These values are however in the allowable range for process j, because $|(\alpha-2\varphi)-\beta|\le\varphi$ Accordingly, the following voting multisets are formed:

$$\begin{aligned}\mathbf{V}_i &= \langle \alpha-\varphi, \alpha, \ldots, \alpha, \alpha+\varphi, \ldots, \alpha+\varphi\rangle & (3.12)\\ \mathbf{V}_j &= \langle \alpha-2\varphi, \ldots, \alpha-2\varphi, \alpha-\varphi, \ldots, \alpha-\varphi\rangle & (3.13)\end{aligned}$$

Similar to Theorem 2, in (3.12) and (3.13), let the elements with values $(\alpha-2\varphi)$, $(\alpha-\varphi)$, α, and $(\alpha+\varphi)$ carry the weights $0, \ldots, 3$, respectively. Using these weights, $\mathbf{V}_i$ and $\mathbf{V}_j$ can be converted into the following enumerated lists:

$$\begin{aligned}E_i\_Sel_\sigma &= \langle 1, 2, \ldots, 2, 3, \ldots, 3\rangle & (3.14)\\ E_j\_Sel_\sigma &= \langle 0, \ldots, 0, 1, \ldots, 1\rangle & (3.15)\end{aligned}$$

In (3.14), there are t elements with weight 3, and there are $(n-t-1)$ elements with weight 2. Similarly, in (3.15), there are t elements with weight 0, and $(n-t)$ elements with weight 1. Again, the weights in (3.10) are merely a different representation of (3.14) and (3.15) respectively. □

3.3 Fault-Tolerance of Asynchronous Egoestic algorithms

By (3.3), convergence occurs if $\sigma > \omega_{\gamma_{2t}}$. To determine the lower bound of N for which a voting algorithm exists, consider the case when all elements from $\mathbf{V}$ are selected, so that $\sigma=n$. Then:

$$C = \frac{\omega_{\gamma_{2t}}}{\sigma} = \frac{\sum_{g=1}^{\gamma_{2t}} \left[e_{i,n-g+1} - e_{j,g}\right]}{n} \quad (3.16)$$

For Asynchronous MSE, the selection of every element implies $k(g)=g$, for $g\le n$. Thus, using (3.4) and

(3.5), $[e_{i,n-g+1} - e_{j,g}] =$

$$\begin{cases} 3-0=3 & : 1 \le g \le a \\ 3-1=2 & : a < g \le (a+s) \\ 2-1=1 & : a+s < g \le 2(a+s) \end{cases} \quad (3.17)$$

In addition, since $\Delta k(g, g+2t) = 2t, \ \forall\ g \le \sigma - 2t$, it follows that $\gamma_{2t} = 2t = 2(a+s)$. Thus, using (3.17), (3.16) becomes:

$$\begin{aligned} C &= \frac{\sum_{g=1}^{a} [e_{i,n-g+1} - e_{j,g}]}{n} + \\ &\quad \frac{\sum_{g=a+1}^{(a+s)} [e_{i,n-g+1} - e_{j,g}]}{n} + \\ &\quad \frac{\sum_{g=a+s+1}^{2(a+s)} [e_{i,n-g+1} - e_{j,g}]}{n} \\ &= \frac{3a + 2s + (a+s)}{n} = \frac{4a+3s}{n} \end{aligned}$$

Accordingly, $n \ge (4a + 3s + 1)$ ensures that $C < 1$. Since $N = n + t$,

$$N \ge (4a + 3s + 1) + t = 5a + 4s + 1$$

This is the same fault-tolerance obtained by a different family of asynchronous voting algorithms called *Mean-Subsequence-Reduced* (MSR) [5].

For Asynchronous MSEP algorithms, the selection of every element implies $k(g) = g,\ g \le n$. Therefore, using (3.10) and (3.11):

$$[e_{i,n-g+1} - e_{j,g}] = \begin{cases} 3-0=3: & 1 \le g \le (a+s) \\ 2-1=1: & a+s < g \le 2(a+s) \end{cases}$$

In addition, since $\Delta k(g, g+2t) = 2t, \ \forall\ g \le \sigma - 2t$, it follows that $\gamma_{2t} = 2t = 2(a+s)$. Thus, (3.16) becomes:

$$\begin{aligned} C &= \frac{\sum_{g=1}^{(a+s)} [e_{i,n-g+1} - e_{j,g}]}{n} + \\ &\quad \frac{\sum_{g=a+s+1}^{2(a+s)} [e_{i,n-g+1} - e_{j,g}]}{n} \\ &= \frac{3(a+s) + (a+s)}{n} = \frac{4(a+s)}{n} \end{aligned}$$

Accordingly, $n \ge (4a + 4s + 1)$ ensures that $C < 1$. Thus,

$$N \ge (4a + 4s + 1) + t = 5a + 5s + 1 = 5t + 1$$

This result shows that the hybrid model of asynchronous MSEP performs the same as the single Byzantine fault model [5], because the effect of symmetric faulty processes is the same as that for asymmetric faults.

4 Working Examples

This section shows two examples on how the convergence rate is obtained. The first example uses the common Fault Tolerant Midpoint [6, 7] voting algorithm. The second example is about voting algorithms that select all odd-numbered elements from the voting multisets. Since the process is the same for both asynchronous MSE and asynchronous MSEP, let us use the process for asynchronous MSE.

Fault-Tolerant Midpoint – Fault-Tolerant Midpoint selects only the two extreme values of its voting multiset values v_1 and v_n. Thus, $\sigma = 2$ and $\gamma_{2t} = 1$. The selection of v_1 and v_n implies that $k(\sigma) = k(2) = n$ and $k(1) = 1$, so that $e_{i,\sigma} = 3$ and $e_{j,1} = 0$. As a result:

$$\omega_{\gamma_{2t}} = \sum_{g=1}^{\gamma_{2t}} [e_{i,\sigma-g+1} - e_{j,g}] = e_{i,\sigma} - e_{j,1} = 3$$

Since $\sigma < \omega_{\gamma_{2t}}$, we have $C > 1$. Consequently, the Fault-Tolerant Midpoint is not convergent under the asynchronous MSEP model. But let us instead select the extreme values beyond the largest and the smallest $(a+s)$ elements, i.e. $v_{(a+s+1)}$ and $v_{n-(a+s)}$. Since $k(\sigma) = k(2) = n - (a+s)$, according to (3.4), when $g = 1$, $e_{i,\sigma-g+1} = e_{i,2} = 2$. Similarly, since $k(1) = (a+s+1)$, when $g = 1$, $e_{j,g} = 1$. Therefore,

$$\omega_{\gamma_{2t}} = \sum_{g=1}^{\gamma_{2t}} [e_{i,\sigma-g+1} - e_{j,g}] = e_{i,\sigma} - e_{j,1} = 1$$

As a result, $C = \omega_{\gamma_{2t}}/\sigma = 1/2$. This is the convergence rate of Fault-Tolerant Midpoint previously obtained under Dolev's Byzantine and Kieckhafer's MSR fault-models [6, 7].

Selecting Odd-numbered Elements – Assume the selection function selects all odd-numbered elements from $\mathbf{V}$. Furthermore, assume $a = 1$, $s = 2$, and $n = 13$, so that: $\sigma = 7$ and $n \ge 4a + 3s + 1 = 11$.

To obtain γ_{2t}, based on its definition, we need to find the smallest value $(h - g)$ such that $\Delta k(g,h) = k(h) - k(g) \ge 2t = 2(a+s) = 6$, for all valid values of g and h. The smallest such value is 3. In other words, if $h = g + 3$, then $\Delta k(g,h) \ge 6 \ \ \forall\ g \in \{1, \ldots, \sigma - 3\}$. Therefore, $\gamma_{2t} = \gamma_6 = 3$. On the other hand, the expressions in (3.4) and (3.5) yield:

$$\begin{array}{ll} e_{i,1} = 1 & e_{j,1} = 0 \\ e_{i,2} = 2 & e_{j,2} = 1 \\ e_{i,3} = 2 & e_{j,3} = 1 \\ e_{i,4} = 2 & e_{j,4} = 1 \\ e_{i,5} = 3 & e_{j,5} = 1 \\ e_{i,6} = 3 & e_{j,6} = 1 \\ e_{i,7} = 3 & e_{j,7} = 1 \end{array}$$

Therefore, the convergence rate is:

$$\begin{aligned} C = \frac{\omega_{\gamma_6}}{\sigma} &= \frac{\sum_{g=1}^{\gamma_6} [e_{i,\sigma-g+1} - e_{j,g}]}{\sigma} \\ &= \frac{\sum_{g=1}^{3} [e_{i,7-g} - e_{j,g}]}{7} = \frac{6}{7} = 0.85 \end{aligned}$$

In comparison, asynchronous MSR algorithms [5] discard the largest and smallest t elements before using the selection function. Discarding these elements leaves 7 elements. The selection function then selects the odd-numbered elements, making $\sigma = 4$. In addition, it is shown [5] that the asynchronous MSR algorithms have the convergence rate:

$$C = \frac{\gamma_{(2a+s)}}{\sigma}$$

As a result:

$$C = \frac{\gamma_{(2a+s)}}{\sigma} = \frac{2}{4} = 0.50$$

Thus, asynchronous MSR algorithms perform better than asynchronous MSE. However, as shown in the previous example, the performance of Egoestic algorithms is much improved if the t extreme elements are not in the selected multiset.

5 Summary and Future Research

This paper has examined the problem of reaching Approximate Agreement for Asynchronous Egoestic voting algorithms. The following remarks can be made:

- Theorems 1-3 show that the fault-tolerance and the convergence rate of any asynchronous Egoestic algorithm can be obtained easily, *without analyzing it individually.* To know about the performance of any of these algorithms, one only needs to know about the selection function and the number of elements that it selects.
- Since better performance on C requires lower values on $[e_{i,n-g+1} - e_{j,g}]$, Theorems 2 and 3, and the working examples, showed that much better performance is achieved if the extreme t elements of the voting multisets are not used by the selection function.
- It has been shown that the fault tolerance of Asynchronous MSE is the same compared to other Asynchronous voting algorithms such as MSR. However the fault tolerance of Asynchronous MSEP is lower than Asynchronous MSE and MSR algorithms.

The studies on Egoestic voting algorithms assume the extreme values as the default values when it comes to replacing a detected erroneous value. It would be interesting to find out what the effect on convergence rate is if one can somehow use a default value that is neither of the extreme values. Or if the voting function can dynamically update the default value based on the knowledge gained from the so far received values.

This paper has furnished the last missing piece in the synchronous and asynchronous communications for the three categories of voting algorithms MSR, MSE, and MSEP, for a total of six different voting environments. Thus, the completion of this work opens the door for the development of a global model encompassing all six forms of voting environments.

References

[1] A. Avizienis, "The N-Version Approach to Fault-Tolerant Software", *IEEE Trans. Software Engineering*, Vol. SE-11, No. 12, Dec 1985, pp. 1491-1501.

[2] M.H. Azadmanesh, A.W. Krings, "Asynchronous Behavior of Egoestic Voting Algorithms", University of Nebraska at Omaha, Computer Science Tec. Rep. UNO-CS-TR-00-01, 2000.

[3] M.H. Azadmanesh, L. Zhou, D.A. Peak, "Egophobic Voting Algorithms", University of Nebraska at Omaha, Computer Science Tec. Rep. UNO-CS-TR-99-01, 1999.

[4] M.H. Azadmanesh, A.W. Krings, "Egocentric Voting Algorithms", *IEEE Trans. Reliability*, Vol. 46, No. 4, Dec 1997, pp. 494-502.

[5] M.H. Azadmanesh, R.M. Kieckhafer, "New Hybrid Fault Models for Asynchronous Approximate Agreement", *IEEE Trans. Computers*, Vol. 45, No. 4, Apr 1996, pp. 439-449.

[6] D. Dolev, et al, "Reaching Approximate Agreement in the Presence of Faults," *JACM*, Vol. 33, No. 3, Jul 1986, pp. 499-516.

[7] R.M. Kieckhafer, M.H. Azadmanesh, "Reaching Approximate Agreement With Mixed Mode Faults", *IEEE Trans. on Parallel Distributed Systems*, Vol. 5, No. 1, Jan 1994, pp. 53-63.

[8] L. Lamport, P.M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults", *JACM*, Vol. 32, No. 1, Jan 1985, pp. 52-78.