EGOPHOBIC VOTING ALGORITHMS

Azad Azadmanesh¹, Lingjie Zhou², Daniel Peak³

Abstract – Approximate Agreement is one form of distributed agreement, in which the processes by executing a voting algorithm are required to agree on values that are very close to each other. Recent studies have partitioned the voting algorithms into three broad categories: Anonymous, Egocentric, Egophobic. Among these categories, Egophobic algorithms have not been studied yet. Hence, this paper considers one family of Egophobic algorithms. They are studied under a hybrid fault model consisting of asymmetric, symmetric, and benign faults. We obtain their Convergence Rate and Fault Tolerance expressions, show that these algorithms can not discriminate between asymmetric and symmetric faults, and compare their performance against the algorithms in other categories. The study of Egophobic algorithms will further research into forming a unified model capable of addressing any algorithms in these categories.

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

¹This work was supported by the NASA Space Grant Consortium and by the Information Science and Technology (IS&T) college of the University of Nebraska at Omaha (UNO).
²UNO, Dept. of Computer Science, Omaha, Ne 68182-0500; E-mail: azad@alvaz.ist.unomaha.edu
³UNO, Dept. of Computer Science, Omaha, Ne 68182-0500; E-mail: linjie@hotmail.com

University of North Texas, Dept. of Business Computer Information Systems, Denton, Texas 76203-5249; E-mail: peak@cobaf.coba.unt.edu
1 Introduction

Many applications require the nodes\(^4\) in a distributed system to reach agreement on a particular event or value, such as the Commit protocol in a distributed database system, or electing a leader in a network. In contrast to this form of agreement where the input and output values held by the nodes are discrete, the nodes in a system may start with non-discrete values, and are supposed to eventually decide on output values within a small positive real-value of each other. Some application areas of such agreement are the management of redundant sensor data and fault-tolerant clock synchronization [1, 2, 3, 4, 5]. This form of agreement, called *approximate agreement*, must satisfy the following two conditions [6, 7]:

**Agreement** – The voted values of any pair of non-faulty processes are within \(\epsilon\) of each other.

**Validity** – The voted value for each non-faulty process is strictly within the range of the initial values of the non-faulty processes.

Traditionally, studies in Approximate Agreement assume faults behave as worst-case Byzantine faults. However, in real world systems, the vast majority of faults do not behave in Byzantine manner [7, 8, 9, 10, 11, 12]. Hence, the results tended to be overly conservative, leading to unnecessarily complex system designs. Recently, to remedy this difficulty, some studies have used hybrid-fault models. These studies include the work done by Fekete [13], and Kieckhafer and Azadmanesh [7, 8, 14].

Fekete [13] analyzed the behavior of omission and Byzantine faults. However, this study is not based on a true hybrid fault model. Specifically, it considers three separate single-mode fault models, each dealing with a different failure mode (crash-failure, failure-by-omission, or Byzantine).

In a different study, Kieckhafer and Azadmanesh [7] used a hybrid fault model which allows for simultaneous existence of three-failure modes (Benign, Symmetric, and Asymmetric).

\(^4\)Henceforth, the words “node” and “process” are used interchangeably.
They presented a methodology to measure the performance of any voting algorithm within a family of voting algorithms called *Mean-Subsequence-Reduced* (MSR). This work was later extended to include the class of omission failure [8, 14]. In a related work, Azadmanesh and Krings [15] used the three failure mode model for a different family of voting algorithms which belongs to the class of *Egocentric* algorithms [2, 16, 17]. In this family of 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. This article investigates the counterpart form of these algorithms, called *Egophobic* algorithms, where the processes place more trust on values other than their own. These algorithms are often used for hardware clock synchronization [18, 19, 20]. In general, Egophobic algorithms, as well as Egocentric algorithms, can be used in applications where the processes may not all produce exactly the same value. Thus, a certain amount of discrepancy among processes must be tolerated. This discrepancy amount depends on the application. The discrepancy might be, for instance, due to sensors reading the same input, 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 [2, 4, 5, 21, 22].

As no general analysis of Egophobic algorithms exists in literature, this study will derive the performance and fault tolerance for such algorithms. In addition, the analysis will be under the same hybrid fault model used for Egocentric algorithms. The treatment of Egophobic algorithms under the same fault model will allow for easier comparison of such algorithms against Egocentric and MSR algorithms.

We analyze the extreme case of Egophobic algorithms, that is, each node replaces the erroneous values by a value farthest from its own. As a result, regardless of the value used to replace erroneous values, we will be able to present the range of performance for any Egoestic (Egocentric or Egophobic) algorithm.

Section 2 provides some background and definitions used throughout the article. Section 3 provides the categories of algorithms and describes their differences. Section 4 explains
the effect of different failure modes on Egophbic algorithms. Section 5 provides some preliminary lemmas. The results of Sections 4 and 5 are then used in Section 6 which obtains the worst and optimal convergence rate and the fault tolerance for these algorithms. Section 7 compares these algorithms against other voting algorithms. Section 8 concludes the paper with a summary and some directions for future research.

2 Background and Definitions

2.1 System Model

We consider completely connected networks consisting of \( N \) nodes, and assume the timing of communication is \emph{synchronous}. In a synchronous communication system, the processing and communication delay of non-faulty processes are bounded. There is thus a point in time when every process will have received all data from non-faulty processes before it executes the voting algorithm. Any data arriving after this time is considered to be from a faulty process.

2.2 Fault Model

We use the hybrid fault model of Thambidurai and Park [11]. This model partitions the total number of faults \( t \) in the system into \( b \) \emph{benign}, \( s \) \emph{symmetric}, and \( a \) \emph{asymmetric} faults. Thus, \( t = a + s + b \). Benign faults, also called manifest faults, are self-incriminating or self-evident to every non-faulty process. Examples of benign faults include out of bound data, data received outside of the bounded communication delay, and crash faults. A symmetric fault is defined as a fault whose erroneous value is perceived identically by all receiving non-faulty processes. An asymmetric fault occurs when the non-faulty processes receive conflicting messages, due to either faults in the communication network, or the sending process maliciously transmitting conflicting values to non-faulty processes.
2.3 Real–Valued Multisets

Approximate Agreement requires the manipulation of multisets of real values. A multiset is a collection of objects similar in concept to a set. However, it differs from a set in that all elements of a multiset are not necessarily distinct. A multiset of real numbers can be represented as a monotonically increasing sequence of the real values of its elements, that is, \( V = \langle v_1, \ldots, v_V \rangle \) is ordered such that: \( v_i \leq v_{i+1} \forall i \in \{1, \ldots, V - 1\} \) [23, 24]. The size of \( V \) is \( V = |V| \). The following operations are useful on multisets:

\[ \rho(V) = [\min(V), \max(V)] = [v_1, v_V]; \text{ the real interval spanned by } V. \rho(V) \text{ is called the range of } V. \]

\[ \delta(V) = \max(V) - \min(V) = v_V - v_1; \text{ the difference between the maximum and minimum values of } V. \delta(V) \text{ is called the diameter of } V. \]

\[ \text{mean}(V) = \text{The arithmetic mean of the real values of all elements of } V. \]

Subsequences – Consider two non-empty multisets \( U \) and \( V \), where \( U \subseteq V \). \( U \) is a subsequence of \( V \) if there is an order-preserving one-to-one mapping \( k \), from the indices of \( U \) to the indices of \( V \), i.e. \( u_j = v_{k(j)} \forall j \in \{1, \ldots, U\} \) and \( k(j) < k(j + 1) \forall j \in \{1, \ldots, U - 1\} \).

Dominance – Consider two equal-sized non-empty multisets \( W = \langle w_1, \ldots, w_W \rangle \) and \( W' = \langle w'_1, \ldots, w'_W \rangle \). \( W \) dominates \( W' \) if and only if \( w_i \geq w'_i \forall i \in \{1, \ldots, W\} \). Dominance has the property that it is preserved under the subsequence operation, that is, given two equal sized multisets \( V \) and \( V' \), let \( U \) be a subsequence of \( V \), and \( U' \) be the same subsequence of \( V' \). If \( V \) dominates \( V' \), then \( U \) dominates \( U' \).

2.4 Single-Step Convergence

During the course of the agreement process, each non-faulty process \( i \) executes the same voting algorithm in rounds as follows. Initially process \( i \) broadcasts its initial value to all processes including itself. Then it collects all the values it has received into a multiset \( N_i \). If process \( i \) does not receive a value from some process, or if it detects that a received
value is erroneous, process \( i \) simply picks some default value to assign to that process in the multiset. The final multiset is called the voting multiset \( V_i \). Process \( i \) then applies a function \( F \) to \( V_i \) 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. The objective of Approximate Agreement is achieved if it is guaranteed that the diameter of the voted values among the correct processes become smaller after each round of voting. This property, called single-step-convergence, ensures that the diameter of the correct values will be eventually less than \( \epsilon \).

To formally define this property, define \( U_{all} \) as the multiset of all correct values received by non-faulty processes. Then, an algorithm is single-step-convergent if both of the following conditions are true following every round of voting:

**Validity** – For each non-faulty process \( i \), the voted value is within the range of correct values, that is, \( F(V_i) \in \rho(U_{all}) \).

**Convergence** – 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, that is, \( |F(V_i) - F(V_j)| < \delta(U_{all}) \).

The effectiveness of a convergent voting algorithm is measured by its convergence rate \( C \). Assuming that \( \delta(U_{all}) > 0 \), \( C \) is the maximum possible value of the ratio:

\[
C = \frac{|F(V_i) - F(V_j)|}{\delta(U_{all})}
\]

If \( C < 1 \) in each round, it is then guaranteed that after enough rounds, the system will achieve the Agreement condition, i.e. the diameter of correct values held by the non-faulty processes in the system will be within \( \epsilon \).
3 Categories of Voting Algorithms

Known voting algorithms can be partitioned into three broad categories: Anonymous, Egocentric, and Egophobic. Fig. 1 shows three families of these algorithms, which are called MSR, MSE, and MSEP. There might be other categories of algorithms which are shown in the figure as Others.

For MSR algorithms, each process uses the following function [7]:

$$F(V_i) = \text{mean} \left[ Sel_\sigma (Red^\tau (V_i)) \right]$$

The Reduction function $Red^\tau$ removes the $\tau$ largest and the $\tau$ smallest elements from multiset $V_i$. $\tau$ is the number of faults to be tolerated. The Selection function $Sel_\sigma$ selects a submultiset of $\sigma$ elements from the reduced multiset, that is, from the multiset generated by $Red^\tau(V_i)$. $F(V_i)$ is the mean of these selected elements.

MSR algorithms can use any value as a default. Therefore, the default value is inherently erroneous. On the other hand, the Mean-Subsequence-Not-Reduced (MSNR) algorithms, which contain the Egocentric and Egophobic categories, attempt to replace a missing value or a detected error with a value within the range of correct values. These algorithms use the following approximation function:

$$F(V_i) = \text{mean} \left[ Sel_\sigma (V_i) \right]$$
For a process $i$ to detect that a value in $N_i$ is erroneous, the value must be apart from the process’s own value by more than a pre-defined tolerance range $\varphi$. Otherwise the value is accepted as a correct value, even if it is generated by a faulty process. Therefore MSNR algorithms are distinguished from MSR algorithms in two aspects:

- MSNR algorithms do not use the Reduction function.
- The range of the initial correct values must be known a-priori by all processes.

Because of this second aspect, MSNR algorithms do not need to use the Reduction function, whose sole purpose is to generate a submultiset whose range is always within the range of correct values. Thus, if $\alpha$ and $\beta$ are the values of two arbitrary non-faulty processes, $\varphi$ is the diameter of correct values, and $v_{i,h}$ and $v_{j,l}$ are the $h^{th}$ and $l^{th}$ values in the voting multisets $V_i$ and $V_j$ respectively, then these algorithms have the following properties:

$$|\alpha - \beta| \leq \varphi$$

$$|v_{i,h} - \alpha| \leq \varphi,$$

$$|v_{j,l} - \beta| \leq \varphi.$$ 

Depending on how one chooses the default value, two families of MSNR algorithms have been identified. They are called MSE and MSEP. In MSE algorithms [15], each process uses its own value as the default value. In other words, a non-faulty process prefers to use its own value for bad or missing values. This work completes the performance analysis of MSNR algorithms by concentrating on MSEP algorithms. For MSEP algorithms, each process prefers to use any default value other than its own value. Although this may not seem to be a big difference in comparison to MSE algorithms (as the default value is still within the range of the correct values), it will be shown that MSEP algorithms need a higher lower bound on the minimum number of processes required to achieve convergence, and have lower performance rate than Egocentric algorithms.

An important parameter to MSNR algorithms is $\gamma$. The purpose of $\gamma$ is to show whether a voting algorithm is convergent. If so, it will determine the convergence rate. (The merits of
\( \gamma \) will be clear by Theorem 1 and the convergence rate expression. The following describes how \( \gamma \) is obtained.

**Definition of \( \gamma \)** — The selected multiset \( S = \text{Sel}_\sigma(V) = \langle s_1, \ldots, s_\sigma \rangle \) is a subsequence of \( V = \langle v_1, \ldots, v_n \rangle \). Thus, each element of \( S \) corresponds to one unique element of \( V \). Now, let \( g \) be the index of any element of \( S \), and let \( k(g) \) be the index of the corresponding element in \( V \), so that \( s_g = v_{k(g)} \), for each \( g \in \{1, \ldots, \sigma\} \).

Given two indices into \( S \), \( g \) 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 \( V \) spanned by elements \((s_g, \ldots, s_h)\) in \( S \). Thus, \( \Delta k(g, h) \) is the number of elements of \( V \) in the submultiset \( \langle v_{k(g)+1}, \ldots, v_{k(h)} \rangle \).

Finally, for any non-negative integer \( z \), \( \gamma_z \) is defined as the minimum value of \((h - g)\) which ensures that \( z \) elements of \( V \) are spanned, independent of \( g \) and \( h \), that is, \( \gamma_z \) is the minimum value of \((h - g)\) that ensures that \( k(h) - k(g) \geq z \), for any values \( g \) and \( h \), \( g \leq h \leq \sigma \). By this definition, \( \gamma_z \) exists only if \( |V| > z \).

The following pseudo-code shows how \( \gamma_z \) is obtained:

\[
\begin{align*}
\gamma_z &\leftarrow -1, \quad I \leftarrow 0 \\
\text{WHILE (} I \leq \sigma - 1 \text{ AND } \gamma_z = -1 \text{ )} &
\{ \\
\quad \text{IF ( for every } g \in \{1, \ldots, \sigma - I\}, \quad \Delta k(g, g + I) \geq z \text{ )}
\quad \gamma_z \leftarrow I \\
\quad I \leftarrow I + 1 \\
\} \\
\end{align*}
\]

By definition, \( \gamma_z \) can not be negative. Thus, \( \gamma_z \) does not exist if it is negative at the end of the pseudo-code.
4 MSEP Voting Algorithms

During a voting procedure, messages may be lost, delayed, or corrupted. Thus a process needs to pick up a default value to replace with these messages. The default value used by Egophobic algorithms is a value other than the process’s own value. Thus, a process, in a sense, places more trust on other processes’ values [15, 17, 18]. Among infinite number of acceptable default values that a process \(i\) can choose, i.e. from the range \((\alpha - \varphi)\) to \((\alpha + \varphi)\), the minimum and the maximum acceptable default values, i.e. \((\alpha - \varphi)\) and \((\alpha + \varphi)\), are most distinguishable. MSEP algorithms use either the minimum or the maximum values for their default values. It will be shown that, no Egophobic algorithm can perform worse than MSEP algorithms. Since the discussion of these two cases are symmetric, in this paper, we just focus on the case which uses the maximum acceptable values as a default, i.e. the sum of the process’s own value and the predefined tolerance value \(\varphi\).

4.1 Fault-Mode Analysis

4.1.1 Benign Faults

A large percentage of the faults occurring within distributed systems are benign faults. Since benign faults are self-evident to all non-faulty processes, the obvious solution is to simply ignore them. Thus the voting algorithms are executed with no benign values in the voting multisets [6, 7]. Thus, it is only necessary to address convergence using the multiset of values not containing benign errors. Hence, the size of \(V\) is \(n = N - b\).

4.1.2 Symmetric Faults

Recall that \(v_{i,h}\) and \(v_{j,l}\) are the \(h^{th}\) and \(l^{th}\) values in the voting multisets \(V_i\) and \(V_j\) respectively. Assume that \(v_{i,h}\) and \(v_{j,l}\) are the values sent by a process \(k\) to processes \(i\) and \(j\) respectively. Furthermore, assume process \(k\) is symmetrically faulty. By definition, the default values for process \(i\) is \((\alpha + \varphi)\) and for process \(j\) is \((\beta + \varphi)\).

If \(v_{i,h}\) and \(v_{j,l}\) are both in the selected multiset of \(i\) and \(j\), then there are four cases to
consider, in order to find the max$|F(V_i) - F(V_j)|$:

a. If $|v_{i,h} - \alpha| \leq \varphi$ and $|v_{j,l} - \beta| \leq \varphi$, both $v_{i,h}$ and $v_{j,l}$ are within the tolerance range, and they are thus accepted by processes $i$ and $j$. Since process $k$ is symmetrically faulty:

$$|v_{i,h} - v_{j,l}| = 0$$  \hspace{1cm} (4.1)

b. If $|v_{i,h} - \alpha| \leq \varphi$ and $|v_{j,l} - \beta| > \varphi$, process $i$ accepts $v_{i,h}$ but process $j$ rejects $v_{j,l}$ as it is outside of the acceptable range. Thus, $v_{j,l}$ is replaced with $(\beta + \varphi)$. Since $\alpha - \varphi \leq v_{i,h} \leq \alpha + \varphi$ and $-\varphi \leq (\alpha - \beta) \leq \varphi$, then:

$$|v_{i,h} - v_{j,l}| = |v_{i,h} - (\beta + \varphi)| \leq |(\alpha - \varphi) - (\beta + \varphi)| = |2\varphi + (\alpha - \beta)| \leq 3\varphi$$

c. If $|v_{i,h} - \alpha| > \varphi$ and $|v_{j,l} - \beta| \leq \varphi$, process $j$ accepts $v_{j,l}$, and process $i$ rejects $v_{i,h}$. Then:

$$|v_{i,h} - v_{j,l}| = |(\alpha + \varphi) - v_{j,l}|$$
$$\leq |(\alpha + \varphi) - (\beta - \varphi)|$$
$$= |2\varphi + (\alpha - \beta)|$$
$$\leq 3\varphi$$

d. If $|v_{i,h} - \alpha| > \varphi$ and $|v_{j,l} - \beta| > \varphi$, process $i$ replaces $v_{i,h}$ with $(\alpha + \varphi)$ and process $j$ replaces $v_{j,l}$ with $(\beta + \varphi)$. Thus:

$$|v_{i,h} - v_{j,l}| = |(\alpha + \varphi) - (\beta + \varphi)|$$
$$= |\alpha - \beta|$$
$$\leq \varphi$$

As a result, among the four cases, cases b and c cause the maximum discrepancy between $i$ and $j$.  

10
4.1.3 Asymmetric Faults

If process $k$ is an asymmetrically faulty process, $v_{i,h}$ and $v_{j,l}$ are not only incorrect but also do not hold the same value. Depending on the behavior of process $k$, four different situations are possible:

a. If $|v_{i,h} - \alpha| \leq \varphi$ and $|v_{j,l} - \beta| \leq \varphi$, both values $v_{i,h}$ and $v_{j,l}$ will be accepted by $i$ and $j$. For this case, the maximum and the minimum acceptable values for $v_{i,h}$ and $v_{j,l}$ are $(\alpha + \varphi)$ and $(\beta - \varphi)$ respectively. Thus:

$$|v_{i,h} - v_{j,l}| \leq |(\alpha + \varphi) - (\beta - \varphi)|$$

$$= |2\varphi + (\alpha - \beta)|$$

$$\leq 3\varphi$$

b. If $|v_{i,h} - \alpha| \leq \varphi$ and $|v_{j,l} - \beta| > \varphi$, process $i$ accepts the value, but $v_{j,l}$ is replaced with $(\beta + \varphi)$. Thus:

$$|v_{i,h} - v_{j,l}| = |v_{i,h} - (\beta + \varphi)|$$

$$\leq |(\alpha - \varphi) - (\beta + \varphi)|$$

$$= |-2\varphi + (\alpha - \beta)|$$

$$\leq 3\varphi$$

c. If $|v_{i,h} - \alpha| > \varphi$ and $|v_{j,l} - \beta| \leq \varphi$, process $i$ replaces $v_{i,h}$ with $(\alpha + \varphi)$, but process $j$ accepts the value $v_{j,l}$. Thus:

$$|v_{i,h} - v_{j,l}| = |(\alpha + \varphi) - v_{j,l}|$$

$$\leq |(\alpha + \varphi) - (\beta - \varphi)|$$

$$= |2\varphi + (\alpha - \beta)|$$

$$\leq 3\varphi$$

d. If $|v_{i,h} - \alpha| > \varphi$ and $|v_{j,l} - \beta| > \varphi$, both values $v_{i,h}$ and $v_{j,l}$ will be replaced with the default values $(\alpha + \varphi)$ and $(\beta + \varphi)$ respectively. Thus:

$$|v_{i,h} - v_{j,l}| = |(\alpha + \varphi) - (\beta + \varphi)|$$
Among the four cases, cases a, b, and c produce the maximum difference.

5 Preliminary Lemmas

**LEMMA 1** : Let \( f \) and \( n \) be non-negative integers, such that \( f \leq n \). Assume an MSEP \( F(V) \), such that \( V = \langle v_1, \ldots, v_n \rangle \) and \( v_i \leq \top \), where \( \top \) is the upper bound of values allowed in \( V \). Assume \( V \) can be partitioned into disjoint multisets \( U = \langle u_1, \ldots, u_{n-f} \rangle \) and \( X = \langle x_1, \ldots, x_f \rangle \) such that \( U \) is a multiset of fixed values and \( X \) is a multiset of variable values. Then, \( F(V) \) generates the maximum possible value if \( x_k = \top \forall x_k \in X \), given \( U \).

**PROOF:** Let \( V = \langle U + X \rangle = \langle u_1, \ldots, u_{n-f}, x_1, \ldots, x_f \rangle \), satisfying the hypothesis \( x_k = \top \forall x_k \in X \). Let \( V' = \langle U + X' \rangle \), where \( X' \) is any other variable multiset other than \( X \) with at least one element less than \( \top \). Let the first such element be \( x'_1 \). Let \( v_g \) be the lowest numbered element in \( V \) for which \( v_g \neq v'_g \). Then, \( v'_g = x'_1 \). The insertion of \( x'_1 \) into \( V' \) shifts the subsequent elements of \( V' \) such that:

\[
\begin{align*}
v_i &= v'_i : 1 \leq i < g \\
v'_i &= v'_i : i = g \\
v'_i &= v'_i : g < i \leq n
\end{align*}
\]

Therefore, \( V \) dominates \( V' \). Because dominance is preserved under subsequence functions, the dominance of \( V \) over \( V' \) implies that \( Sel_\sigma(V) \) dominates \( Sel_\sigma(V') \). As a result \( F(V) \geq F(V') \).

**LEMMA 2** : Let \( f \) and \( n \) be non-negative integers, such that \( f \leq n \). Assume an MSEP \( F(V) \), such that \( V = \langle v_1, \ldots, v_n \rangle \) and \( v_i \geq \bot \), where \( \bot \) is the lower bound of values allowed in \( V \). Assume \( V \) can be partitioned into disjoint multisets \( U = \langle u_1, \ldots, u_{n-f} \rangle \) and \( X = \langle x_1, \ldots, x_f \rangle \) such that \( X \) is a multiset of variable values.

\[
\begin{align*}
\alpha - \beta &= |\alpha - \beta| \\
&\leq \varphi
\end{align*}
\]
\[\langle x_1, \ldots, x_f \rangle\] such that \(U\) is a multiset of fixed values and \(X\) is a multiset of variable values. Then, \(F(V)\) generates the minimum possible value if \(x_k = \bot \forall x_k \in X\), given \(U\).

**PROOF:** The proof is similar to that of Lemma 1.

To determine the convergence rate \(C\), we need to find the maximum difference between two voted values belonging to two arbitrary non-faulty processes \(i\) and \(j\). It is obvious that the maximum difference between \(F(V_i)\) and \(F(V_j)\) is reached when \(F(V_i)\) and \(F(V_j)\) reach their maximum and minimum values respectively.

Furthermore, to find \(C\), one needs only to find a worst-case scenario that maximizes \(|F(V_i) - F(V_j)|\). When we look at the worst case scenarios produced by cases b and c for the symmetric faults and cases a, b, and c for the asymmetric faults, we see that case c is common to both failure modes. Case c indicates that regardless of whether the worst faulty behavior originates from a symmetrically or an asymmetrically faulty process, the maximum discrepancy, i.e. \(3\phi\), is achieved when \(\alpha = \beta + \phi\). For this case, we also observe that process \(i\) replaces the erroneous symmetric value by its default value \((\alpha + \phi)\). As a result, symmetric values are behaving like asymmetric faults, because processes \(i\) and \(j\) ultimately use two different values for a symmetrically faulty process.

According to Lemma 1, all the values of elements in \(U\) are fixed, which can be interpreted as the submultiset of correct values common to both \(i\) and \(j\). Furthermore, based on the same lemma, \(X\) is the submultiset of variable values. These values can be interpreted as the submultiset of erroneous values, because these values may not be common to both processes \(i\) and \(j\). Because \((\alpha + \phi)\) is the largest tolerable value for process \(i\), \(F(V_i)\) is maximized only when all of its elements in \(X\) are equal to \((\alpha + \phi)\). The value \((\alpha + \phi)\) corresponds to case c. Using a similar argument, according to Lemma 2, \(F(V_j)\) will be minimized when all the elements in its submultiset \(X\) are equal to the lowest tolerable value \((\beta - \phi)\). This value again corresponds to case c for both symmetric and asymmetric faults. Based on these observations, the maximum and minimum of \(V_i\) and \(V_j\) are achieved when:

\[
V_i = \langle U_{i \cap j}, A', C' \rangle = \langle \alpha - \phi, \alpha, \ldots, \alpha, \alpha + \phi, \ldots, \alpha + \phi \rangle
\]  
(5.1)
\[ V_j = \langle A, C, U_{ij} \rangle = \langle \alpha - 2\varphi, \ldots, \alpha - 2\varphi, \alpha - \varphi, \ldots, \alpha - \varphi \rangle \]  

(5.2)

where,

\[ U_{ij} = \] the multiset of correct values received by both processes \( i \) and \( j \).

\[ A' = \] the multiset of asymmetric errors used by process \( i \) but not by process \( j \). The multiset contains the default value used by process \( i \), that is, \((\alpha + \varphi)\), for those values received that are outside of the tolerance range. Thus, \(|A'| = a\).

\[ A = \] the multiset of asymmetric errors received by process \( j \). Thus, \(|A| = a\).

\[ C = \] the multiset of symmetric errors received by process \( j \).

\[ C' = \] the multiset of values used by process \( i \) as a result of symmetric faults. This multiset is the same as \( C \) except that the values that are outside of the range are replaced with the default value \((\alpha + \varphi)\) used by process \( i \).

In (5.1), the first \( n - (a + s) \) elements, i.e. \( \langle \alpha - \varphi, \alpha, \ldots, \alpha \rangle \) belong to \( U_{ij} \) of which the first two values \((\alpha - \varphi)\) and \( \alpha \) are held by the processes \( j \) and \( i \) respectively. The rest of the elements, i.e. \( \langle \alpha + \varphi, \ldots, \alpha + \varphi \rangle \), are the asymmetric and symmetric errors whose values have been replaced by their default value \((\alpha + \varphi)\) (see case c). Similarly, the first \((a + s)\) elements of (5.2) having the values \((\alpha - 2\varphi)\) belong to \( A \) and \( C \) (see case c). These are the symmetric and asymmetric values which were transmitted as \((\beta - \varphi)\) and thus were replaced by \( i \) with \((\alpha + \varphi)\). Note that in the worst case \( \beta = \alpha - \varphi \), and thus \( \beta - \varphi = \alpha - 2\varphi \). The rest of the elements in (5.2) belong to \( U_{ij} \). These multisets are referred to as virtual multisets because there are elements in \( V_i \) and \( V_j \) that are treated as correct but are not common to both processes \( i \) and \( j \). For example, the second element of \( V_i \), \( \alpha \), which is the value of process \( i \), is not in \( V_j \). This is to ensure that, regardless of whether \( s_{i,g} \) and/or \( s_{j,g} \) are correct or erroneous, \((s_{i,g} - s_{j,g})\) is maximized.

The following lemma takes advantage of the results in (5.1) and (5.2) to find the maximum of \(|F(V_i) - F(V_j)|\).
LEMMA 3 : Let \( V_i \) and \( V_j \) be two voting multisets of an MSEP voting algorithm. Then:

\[
|F(V_i) - F(V_j)| \leq \frac{I}{\sigma} \left[ \gamma(\alpha+\varepsilon) \sum_{g=1}^{\sigma} (s_{i,\sigma-g+1} - s_{j,g}) \right]
\] (5.3)

PROOF: According to (5.1) and (5.2),

\[ V_i = \langle U_{i \cap j}, A', C' \rangle \]
\[ V_j = \langle A, C, U_{i \cap j} \rangle \]

Now let us define \( W = A + C + U_{i \cap j} + A' + C' \) which has the complete ordering:

\[ W = \langle A, C, U_{i \cap j}, A', C' \rangle \]

Mapping the indices of \( W \) into the indices of the components of \( W \) yields:

\[
\begin{align*}
\langle w_1, \ldots, w_a \rangle &= A = \langle a_1, \ldots, a_a \rangle \\
\langle w_{a+1}, \ldots, w_{a+s} \rangle &= C = \langle c_1, \ldots, c_s \rangle \\
\langle w_{a+s+1}, \ldots, w_n \rangle &= U_{i \cap j} = \langle u_1, \ldots, u_{n-a-s} \rangle \\
\langle w_{n+1}, \ldots, w_{n+a} \rangle &= A' = \langle a'_1, \ldots, a'_a \rangle \\
\langle w_{n+a+1}, \ldots, w_{n+a+s} \rangle &= C' = \langle c'_1, \ldots, c'_s \rangle
\end{align*}
\]

Applying these components to (5.4) produces:

\[
\begin{align*}
V_i &= \langle w_{a+s+1}, \ldots, w_{n+a+s} \rangle \\
V_j &= \langle w_1, \ldots, w_n \rangle
\end{align*}
\]

Thus, the selected multisets \( S_i \) and \( S_j \) can be written as:

\[
\begin{align*}
S_i &= \langle s_{i,1}, \ldots, s_{i,\sigma} \rangle = Sel_{\sigma}(V_i) = Sel_{\sigma}(\langle w_{a+s+1}, \ldots, w_{n+a+s} \rangle) \quad (5.5) \\
S_j &= \langle s_{j,1}, \ldots, s_{j,\sigma} \rangle = Sel_{\sigma}(V_j) = Sel_{\sigma}(\langle w_1, \ldots, w_n \rangle) \quad (5.6)
\end{align*}
\]

Now consider the elements \( s_{i,g} \) and \( s_{j,h} \) of (5.5) and (5.6) respectively:

\[
\begin{align*}
s_{i,g} &= v_{i,k(g)} = w_{a+s+k(g)} \\
s_{j,h} &= v_{j,k(h)} = w_{k(h)} = w_{\Delta k(g,h)+k(g)}
\end{align*}
\]
If $\Delta k(g, h) \geq (a + s)$, then $s_{j, h} \geq s_{i, g}$. Thus, based on the definition of $\gamma_z$, by replacing $h$ with $g + \gamma(a+s)$, $\Delta k(g, g + \gamma(a+s)) \geq (a + s)$. Hence:

$$s_{j, g + \gamma(a+s)} \geq s_{i, g}, \quad 1 \leq g \leq (\sigma - \gamma(a+s))$$

As a result:

$$\sum_{g=\gamma(a+s)+1}^{\sigma} s_{j, g} \geq \sum_{g=1}^{\sigma - \gamma(a+s)} s_{i, g}$$

(5.7)

Using (5.7), $S_i$ and $S_j$ can now be expanded into:

$$S_i = \langle s_{i,1}, \ldots, s_{i,\sigma - \gamma(a+s)}, s_{i,\sigma - \gamma(a+s)+1}, \ldots, s_{i,\sigma} \rangle$$

(5.8)

$$S_j = \langle s_{j,1}, \ldots, s_{j,\gamma(a+s)}, s_{j,\gamma(a+s)+1}, \ldots, s_{\sigma} \rangle$$

(5.9)

For readability sake, the subscript of $\gamma$ will not be shown, but $\gamma(a+s)$ will be implied. Now, without loss of generality assume that $F(V_i) \geq F(V_j)$. Using (5.8) and (5.9):

$$|F(V_i) - F(V_j)| = F(V_i) - F(V_j)$$

$$= \frac{\sum_{g=1}^{\sigma} s_{i,g}}{\sigma} - \frac{\sum_{h=1}^{\sigma} s_{j,h}}{\sigma}$$

$$= \frac{\sum_{g=1}^{\sigma - \gamma} s_{i,g} + \sum_{g=\gamma+1}^{\sigma} s_{i,g}}{\sigma} - \frac{\sum_{h=1}^{\gamma} s_{i,h} + \sum_{h=\gamma+1}^{\sigma} s_{j,h}}{\sigma}$$

$$= \left[ \frac{\sum_{g=1}^{\sigma - \gamma} s_{i,g} - \sum_{h=\gamma+1}^{\sigma} s_{j,h}}{\sigma} \right] + \left[ \frac{\sum_{g=\gamma+1}^{\sigma} s_{i,g} - \sum_{h=1}^{\gamma} s_{j,h}}{\sigma} \right]$$

(5.10)

Due to (5.7), $\sum_{h=\gamma+1}^{\sigma} s_{j,h} \geq \sum_{g=1}^{\sigma - \gamma} s_{i,g}$. Therefore, the maximum of (5.10) is obtained when $\sum_{h=\gamma+1}^{\sigma} s_{j,h}$ holds its lower bound value, that is, when $\sum_{h=\gamma+1}^{\sigma} s_{j,h} = \sum_{g=1}^{\sigma - \gamma} s_{i,g}$. Hence, (5.10) becomes:

$$|F(V_i) - F(V_j)| \leq \frac{1}{\sigma} \left[ \sum_{g=\gamma+1}^{\sigma} s_{i,g} - \sum_{h=1}^{\gamma} s_{j,h} \right]$$

$$= \frac{1}{\sigma} \left[ \sum_{g=1}^{\gamma} (s_{i,g+\gamma} - s_{j,g}) \right]$$

$$= \frac{1}{\sigma} \left[ \sum_{g=1}^{\gamma} (s_{i,g+\gamma+1} - s_{j,g}) \right]$$

(5.11)

□
In (5.11), depending on the selection function, it is not clear what the value of \((s_{i,\sigma-g+1} - s_{j,g})\) will be. However, according to (5.1) and (5.2), in the worst case, the maximum and minimum of \(s_{i,\sigma-g+1}\) and \(s_{j,g}\) are \((\alpha + \varphi)\) and \((\alpha - 2\varphi)\) respectively. Furthermore, by the indices in (5.11), it is seen that \(s_{i,\sigma-g+1}\) refers to the last \(\gamma\) elements of the selected multiset and \(s_{j,g}\) refers to the first selected elements. This observation is instrumental in obtaining the convergence rate, as is seen below.

To obtain a numerical value for the expression in (5.11), assign weights to the worst case values contained in multisets \(V_i\) and \(V_j\). In (5.1) and (5.2), let the elements with values \((\alpha - 2\varphi)\), \((\alpha - \varphi)\), \(\alpha\), and \((\alpha + \varphi)\) carry the weights \(0, \ldots, 3\), respectively. As an example, in (5.11), if \(s_{j,g} = \alpha - 2\varphi\), then the weight of the selected element is 0. Using these weights, \(V_i\) and \(V_j\) can be converted into the following enumerated lists:

\[
E_i - Sel_{\sigma} = \langle e_{i,1}, \ldots, e_{i,\sigma} \rangle = \langle 1, 2, \ldots, 2, 3, \ldots, 3 \rangle
\]

\[
E_j - Sel_{\sigma} = \langle e_{j,1}, \ldots, e_{j,\sigma} \rangle = \langle 0, \ldots, 0, 1, \ldots, 1 \rangle
\]

where

\[
e_{i,x} = \begin{cases} 
1 & : k(x) = 1 \\
2 & : 2 \leq k(x) \leq (n - a - s) \\
3 & : otherwise 
\end{cases} \quad (5.12)
\]

and

\[
e_{j,y} = \begin{cases} 
0 & : 1 \leq k(y) \leq (a + s) \\
1 & : otherwise 
\end{cases} \quad (5.13)
\]

Now using the sum-expression in (5.11), define the weight function \(\omega_{\gamma_z}\), to be:

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

(5.14)

where the value of \(z\) for the expression in (5.11) is \((a + s)\). For an example, if \(g = 1\) and \(k(\sigma - g + 1) = n\), then \(e_{i,\sigma-g+1} = 3\), which implies that \(s_{i,\sigma-g+1} = v_{i,k(\sigma-g+1)} = v_{i,n}\), which in the worst case is \((\alpha + \varphi)\) (see \(V_i\) in (5.1)). Similarly, if \(g = 1\) and \(k(g) = 1\), then \(e_{j,g} = 0\). In the worst case \(s_{j,g} = v_{j,k(g)} = \beta - \varphi = \alpha - 2\varphi\) (see \(V_j\) in (5.2)). As a result:

\[
s_{i,\sigma-g+1} - s_{j,g} = (\alpha + \varphi) - (\alpha - 2\varphi) = 3\varphi = (e_{i,\sigma-g+1} - e_{j,g})\varphi
\]

(5.15)
6 Performance of MSEP Model

6.1 Convergence Rate

The following theorem takes advantage of the discussion above to obtain the convergence rate.

THEOREM 1: Given an MSEP voting-algorithm $F(V)$,

$$C \leq \frac{\omega (\gamma + \epsilon)}{\sigma}$$

PROOF: Lemma 3 showed that:

$$|F(V_i) - F(V_j)| \leq \frac{1}{\sigma} \left[ \sum_{g=1}^{\gamma} (s_i,\sigma - g + 1 - s_j, g) \right]$$

(6.1)

In the worst case, it was defined that (see (5.15)):

$$\sum_{g=1}^{\gamma} (s_i,\sigma - g + 1 - s_j, g) = \varphi \sum_{g=1}^{\gamma} (e_i,\sigma - g + 1 - e_j, g) = \varphi \omega \gamma$$

(6.2)

Using (6.1) and (6.2),

$$|F(V_i) - F(V_j)| \leq \frac{\varphi \omega \gamma}{\sigma}$$

As a result,

$$C \leq \frac{|F(V_i) - F(V_j)|}{\delta(U_{all})}$$

$$\leq \frac{|F(V_i) - F(V_j)|}{\delta(U_{i \cap j})}$$

$$= \frac{|F(V_i) - F(V_j)|}{\varphi}$$

$$= \frac{\omega \gamma}{\sigma}$$

(6.3)
6.2 Fault-Tolerance

Recall that, for readability, the subscript of $\gamma_{(a+s)}$ is not shown. Theorem 1 showed that $C \leq \omega_\gamma / \sigma$. This relation implies that a voting algorithm is convergent if $\sigma > \omega_\gamma$.

Recall that $N = n + b$. To determine the lower bound of $N$ for which a voting algorithm exists, consider the case when all elements from $V$ are selected, so that $\sigma = n$. The selection of every element implies $k(g) = g$, $g \leq n$. Thus, $\gamma = (a + s)$. Using (5.12) and (5.13), $e_{i,n-g+1} = 3$ and $e_{j,g} = 0$, for $1 \leq g \leq (a + s)$, respectively. Consequently:

$$C = \frac{\omega_\gamma}{\sigma} = \frac{\sum_{g=1}^{\gamma} (e_{i,\sigma-g+1} - e_{j,g})}{\sigma} = \frac{\sum_{g=1}^{\gamma} (e_{i,n-g+1} - e_{j,g})}{n} = \frac{\sum_{g=1}^{(a+s)} (e_{i,n-g+1} - e_{j,g})}{n} = \frac{3(a + s)}{n}$$

Accordingly, $n \geq (3a + 3s + 1)$ ensures that $C < 1$. By incorporating the impact of benign faults, a convergent voting algorithm exists only if:

$$N \geq 3a + 3s + b + 1 \quad (6.4)$$

The lower bound on $N$ for MSR and MSE models are: $N \geq 3a + 2s + b + 1$. Hence, MSEP algorithms have lower fault tolerance. This conclusion is due to the fact that symmetric faults in the worst case behave exactly like the worst case for asymmetric faults. Therefore, these algorithms do not benefit from the Thambidurai and Park fault model [11], that is, partitioning the malicious faults into symmetric and asymmetric fault modes. This is consistent with our fault effectiveness analysis, case c, that both types of faults cause the maximum $3\varphi$ discrepancy between two non-faulty processes.

6.3 Optimal Convergence Rate

THEOREM 2: The optimal convergence rate for any MSEP voting algorithm is:

$$C = \frac{1}{\left[\frac{N-2(a+s)-b-1}{(a+s)}\right] + 1}$$
**PROOF:** To obtain the best performance, we need to find the minimum value for $\omega_{\gamma_z}/\sigma$. This occurs when $\omega_{\gamma_z}$ and $\sigma$ are minimized and maximized respectively. In (5.14), $\omega_{\gamma_z}$ is minimized when $\gamma_z = 1$ and $[e_{i,\sigma-g+1} - e_{j,g}] = 1$. To ensure that $[e_{i,\sigma-g+1} - e_{j,g}] = 1$, $e_{i,\sigma-g+1}$ and $e_{j,g}$ must be either 1 and 0, or 2 and 1, respectively (see (5.12) and (5.13)).

In (5.12), if $e_{i,\sigma-g+1} = 1$, then $k(x) = k(\sigma - g + 1) = 1$, which implies that $\sigma - g + 1 = 1$. Thus, in the expression:

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

(6.5)

when $g = 1$, $\sigma - g + 1 = 1$ indicates that $\sigma = 1$. However, for the algorithm to be convergent, $\sigma$ must be at least 2, otherwise $\omega_{\gamma_z}/\sigma \geq 1$. Therefore, the only choice remaining is for $e_{i,\sigma-g+1}$ and $e_{j,g}$ to be 2 and 1 respectively. In (5.12), when $e_{i,\sigma-g+1} = 2$, $2 \leq k(\sigma - g + 1) \leq n - (a+s)$ indicates that the selected element is not in the last $(a+s)$ elements of $V_i$. Similarly, in (5.13), when $e_{j,g} = 1$, the selected element is not in the first $(a+s)$ elements. As a result, for the expression in (6.5) to evaluate to 1, $\gamma_z$ must be 1 and the selected elements must be in $\langle v_{(a+s)+1}, \ldots, v_{n-(a+s)} \rangle$ of $V$. This submultiset has the length $n - 2(a+s)$.

On the other hand, Kieckhafer and Azadmanesh [7] showed that for any multiset $M$, the maximum number of selected elements, while retaining $\gamma_z = 1$, is $\left\lceil \frac{M-1}{x} \right\rceil + 1$. Thus:

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

$$= \frac{1}{\frac{n-2(a+s)-1}{a+s}} + 1$$

$$= \frac{1}{\frac{N-2(a+s)-b-1}{a+s}} + 1$$

\[\square\]

### 7 Comparison of Voting Models

Several known voting algorithms have been analyzed under both single-mode and hybrid fault models [2, 6, 7, 12, 15, 25]. This section compares the performance of these algo-
gorithms to that of MSEP algorithms. To better understand how to obtain the performance of these algorithms, we explain the process for the Fault-Tolerant Midpoint and Fault-Tolerant Mean. Then, a table is given which summarizes the performance of MSR, MSE, and MSEP algorithms.

**Fault-Tolerant Midpoint** – The Fault-Tolerant Midpoint selects only the two extreme values of its multiset values, i.e. \( v_1 \) and \( v_n \). Thus, \( \sigma = 2 \) and \( \gamma = 1 \). In addition, 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:

\[
\omega_\gamma = \sum_{g=1}^{\gamma} [e_{i,\sigma-g+1} - e_{j,g}] = e_{i,\sigma} - e_{j,1} = 3
\]

As \( \sigma < \omega_\gamma \), \( C > 1 \). Consequently, the Fault-Tolerant Midpoint is not convergent under the MSEP model.

Instead of selecting \( v_1 \) and \( v_n \), however, let us remove the largest and the smallest \((a+s)\) elements and then select the extreme values, i.e. \( v_{(a+s)+1} \) and \( v_{n-(a+s)} \). As \( k(\sigma) = k(2) = n-(a+s) \), according to (5.12), when \( g = 1, e_{i,\sigma-g+1} = e_{i,2} = 2 \). Similarly, as \( k(1) = (a+s+1) \), according to (5.13), when \( g = 1, e_{j,g} = 1 \). Therefore,

\[
\omega_\gamma = \sum_{g=1}^{\gamma} [e_{i,\sigma-g+1} - e_{j,g}] = e_{i,\sigma} - e_{j,1} = 1
\]

As a result, \( C = \omega_\gamma / \sigma = 1/2 \). This is the convergence rate of Fault-Tolerant Midpoint previously obtained under Dolev’s single-mode and Kieckhafer’s MSR fault-models [6, 7, 25].

**Fault-Tolerant Mean** – This algorithm selects all elements of \( V \). Thus, \( \sigma = N - b \).

Furthermore, as \( \Delta k(g, g+1) = 1 \), for \( g \leq (\sigma - 1), \gamma_{(a+s)} = (a+s) \). According to (5.12) and (5.13), \( \sum_{g=1}^{(a+s)} [e_{i,\sigma-g+1} - e_{j,g}] = 3 \), for \( 1 \leq g \leq (a+s) \). Thus:

\[
C = \frac{\omega_{(a+s)}}{\sigma} = \frac{\sum_{g=1}^{(a+s)} [e_{i,\sigma-g+1} - e_{j,g}]}{N - b} = \frac{3(a+s)}{N - b}
\]

When the total number of faults in the system is \( t = a + s + b \),

\[
C = \frac{3(a+s)}{N - b} = \frac{3t - 3b}{N - b}
\]

The single-mode fault model has the convergence rate [6]:

\[
C = \frac{t}{N - 2t} \tag{7.1}
\]
It is observed that the MSEP fault model performs worse than the single-mode fault-model if benigns are not the dominant mode of failure. However, if \((a + s)\) elements from the extreme right and the extreme left are not selected, then \([e_{i,\sigma-g+1} - e_{j,g}] = 1\). Thus:

\[
C = \frac{\omega_{(a+s)}}{\sigma} = \frac{\sum_{g=1}^{(a+s)} [e_{i,\sigma-g+1} - e_{j,g}]}{\sigma} = \frac{a + s}{(N - b) - 2(a + s)} = \frac{t - b}{N - 2t + b}
\] (7.2)

which shows much faster convergence than (7.1). The Fault-Tolerant Mean under the MSR fault model discards the extreme \((a + s)\) elements from both sides of the voting multiset as well. It has the convergence rate [7]:

\[
C = \frac{a}{N - 2t + b}
\] (7.3)

By comparing (7.2) and (7.3), it is observed that, unless \(s = 0\), MSR fault-model shows better performance. The reason for worse performance is that, in the worst case, MSEP can not distinguish between symmetric and asymmetric faults.

Table 1 shows the convergence rates under different fault-models and voting algorithms. The Byzantine fault model is the single mode fault model where every fault is considered Byzantine. MSR, MSE, and MSEP fault models use the hybrid failure modes consisting of asymmetric, symmetric, and benign failure modes, so that \(t = a + s + b\). MSEP\(_1\) stands for the original algorithms, where every element in the voting multiset, which is of size \(N - b\), is a candidate for selection. MSEP\(_2\) stands for those algorithms which do not select any element from the extreme \((a + s)\) elements in the voting multiset. Thus, the size of the submultiset is \(N - 2(a + s) - b\). MSE\(_1\) and MSE\(_2\) are defined similarly [15].

MSEP share some characteristics in performance with MSR and MSE algorithms [7, 15]. But because the effects of symmetric faults do not distinguish themselves from the effects of asymmetric faults, the performance of MSEP algorithms is not better than that of MSR. In comparison to MSE algorithms, it has been shown that MSE algorithms perform their best when no elements from the right \(a + s\) and left \(a\) elements are selected [15]. These algorithms perform better than MSEP\(_1\) and MSEP\(_2\) algorithms.

Among the algorithms that place a bound on the diameter of correct values, i.e. \(|\alpha - \beta| \leq \varphi\), the MSE and MSEP algorithms presented herein use the extreme default values. Specifically,

<table>
<thead>
<tr>
<th>Voting Alg’m</th>
<th>Byzantine</th>
<th>MSR</th>
<th>MSE₁</th>
<th>MSE₂</th>
<th>MSEP₁</th>
<th>MSEP₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-Point</td>
<td>(\frac{t}{2})</td>
<td>(\frac{t}{2})</td>
<td>NC</td>
<td>(\frac{t}{2})</td>
<td>NC</td>
<td>(\frac{t}{2})</td>
</tr>
<tr>
<td>FT-Mean</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{a}{N-2t+b})</td>
<td>(\frac{3a+2s}{N-b})</td>
<td>(\frac{t-b}{N-2t+b})</td>
<td>(\frac{3a+3s}{N-b})</td>
<td>(\frac{t-b}{N-2t+b})</td>
</tr>
<tr>
<td>S-M Opt.</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{3t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{3t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
</tr>
<tr>
<td>M-M Opt.</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{3t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
<td>(\frac{3t}{N-2t})</td>
<td>(\frac{t}{N-2t})</td>
</tr>
</tbody>
</table>

in MSE model, processes use their own values for the default value, whereas MSEP algorithms use the extreme counter-part value, that is, the default value is the process’s own value plus \(\varphi\). As a result, among the non-reduced voting algorithms, regardless of what the chosen default value might be, the performance of Egocentric and Egophobic algorithms will always be within the performance range of MSE and MSEP algorithms.

Table 2 summarizes the minimum number of processes required to guarantee the existence of a convergent voting algorithm. We can see that the benefit of MSEP over the Byzantine fault model is only in the benign faults. Thus, the MSEP is actually a two-mode fault model similar to the fault model of Meyer and Pradhan [9], partitioning the faults into malicious versus non-malicious faults. MSE and MSR have the same fault tolerance, and their minimum requirement on the number of processes is better than the Byzantine fault model.
Table 2: Comparison of Fault Tolerance among Byzantine, MSR, MSE, and MSEP fault models.

<table>
<thead>
<tr>
<th>Fault Model</th>
<th>Byzantine</th>
<th>MSR</th>
<th>MSE</th>
<th>MSEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Tolerance</td>
<td>$N \geq 3t + 1$</td>
<td>$N \geq 3a + 2s + b + 1$</td>
<td>$N \geq 3a + 2s + b + 1$</td>
<td>$N \geq 3a + 3s + b + 1$</td>
</tr>
</tbody>
</table>

8 Summary and Future Research

In some applications, a certain amount of discrepancy among processes must be tolerated. Knowing this discrepancy amount, by following a number of single-step-convergent rounds, MSNR algorithms allow for the processes to further converge their results to within a small positive real value $\epsilon$.

Two extreme-end families of MSNR algorithms are MSE and MSEP. This article has focused on MSEP algorithms. These algorithms, although used in some applications, were never analyzed under a general methodology to better understand their performance, or to be able to compare their performance against other voting algorithms. Therefore this article has derived general expressions that easily determine the performance and fault tolerance of any of these algorithms, rather than analyzing each algorithm individually.

There are several directions in which to expand upon this work. MSEP and MSE algorithms together complete the study of MSNR algorithms. Therefore, it is of benefit to devise a unified model to encompass MSNR and MSR algorithms. Furthermore, MSNR algorithms use the extreme default values. It would be interesting to realize how the performance is affected if the default value is neither of the extreme values, to potentially further reduce the discrepancies among voted values in each round. For example, if the voting function can dynamically update the default value based on the knowledge gained from the so-far received values.

Finally, knowing the values $\epsilon$, $\varphi$, and the worst convergence rate, each process can determine
the number of rounds to converge. As the convergence rate in a round can be better than the worst case convergence rate, it is an open question on finding a methodology to determine the convergence rate in each round dynamically. Such a solution will most likely reduce the number of voting rounds to reach convergence.

References


**Biographies of Authors**

**Azad Azadmanesh** received the BS degree in Cost Accounting from the Institute of Advanced Accounting, and the MS (1982) and PhD (1993) degrees in Computer Science from Iowa State University and University of Nebraska-Lincoln respectively. He is currently an associate professor in the Department of Computer Science at the University of Nebraska at Omaha. His research interests include Fault-Tolerant and Survivable Systems, Distributed Agreement, Reliability Modeling, and Network Communication.

**Lingjie Zhu** received her BS degree in Space Physics from Peking University, China, and MS degree in Physics from University of California, San Diego. She received the MA degree in Computer Science in 1996 from University of Nebraska at Omaha. She is currently a Software Engineer at Plantronics, Inc., Santa Cruz, California. Her specialties are in Windows Programming and Computer Multimedia.

**Daniel Peak** received his Ph.D. in Business Computer Information Systems in 1994 from the University of North Texas (UNT), with majors in Business Computer Information Systems and in Finance. He helped in the development of the University of Nebraska at Omaha’s College of Information Science and Technology. He is currently at the UNT College of Business Administration. His research interests include IT governance, strategic IT planning, media systems and human factors. Dr. Peak has more than 20 years of IT consulting and planning experience with numerous Fortune 500 companies, and has won and participated in numerous production and research grants.