Analysis of a Realistic Fault Model for Large Distributed Systems

M.H. Azadmanesh [1] A.W. Krings [2]

In contrast to the classic form of distributed agreement, called *Byzantine Agreement*, *Approximate Agreement* does not require non-faulty processes to achieve exact agreement. Rather, non-faulty processes need only agree on values within a predefined tolerance.

Recent research has revealed simple expressions for the convergence rate and fault tolerance of a broad family of convergent voting algorithms called *Mean-Subsequenced-Reduced* (MSR) algorithms. The analysis was done for simultaneous presence of *asymmetric*, *symmetric*, and *benign* fault modes. This paper introduces a new fault-model, *Omission-MSR* (OMSR). The new model broadens the applicability of hybrid fault-models by introducing an additional fault-mode, *omissive* faults. It will be shown that OMSR has a number of advantages over the MSR model. Also, the results of the two models will be transformed into a singular set of relations and tables so that their convergence rate and fault tolerance can be compared. In the context of this paper, these relations and tables provide the means to easily determine the convergent properties of different voting algorithms for completely and partially connected networks.

[1]Department of Computer Science and Center for Management of Information Technology, University of Nebraska at Omaha, Omaha, NE 68182–0459 USA.

[2]Department of Computer Science, University of Idaho, Moscow, Idaho 83844-1010 USA.

Contents

1 Introduction

An important issue in fault-tolerant distributed computing is the ability of non-faulty processes to reach agreement on data values in the presence of faulty processes. This issue arises whenever non-faulty processes legitimately form differing "opinions" regarding the correct value. They must then exchange and vote upon their local values to arrive at a single consensus value. The problem is significantly more complex if a faulty process is permitted to send conflicting values to different non-faulty processes. A faulty process with this property has been called two-faced, Byzantine, or asymmetric.

In many real world applications, such as sensor data management and fault-tolerant clock synchronization [19, 24, 26, 29, 31], non-faulty processes need not achieve exact agreement. Rather, they need only agree on a value to within a specified tolerance. This criterion is known as *Approximate Agreement*. Given an arbitrarily small positive real value ϵ, Approximate Agreement is defined by two conditions [13, 15]:

A1: AGREEMENT – The voting algorithms executed by all non-faulty processes eventually halt with voted values that are within ϵ of each other.

A2: VALIDITY – The voted value held by each non-faulty process is within the range of the initial values held by the non-faulty processes.

Recent research has addressed convergent voting in the simultaneous presence of multiple fault modes [20]. This work used the hybrid fault model of Thambidurai and Park, which partitions faults into three modes: asymmetric (Byzantine), symmetric (single-valued) and benign (self-incriminating)[30]. Using this hybrid fault model, simple expressions were derived for the performance and fault-tolerance of a broad family of convergent voting algorithms called *Mean-Subsequence-Reduced* (MSR) algorithms.

MSR algorithms produced more accurate bounds on the properties of the algorithms than possible with any single-mode fault model. However, these algorithms along with other traditional voting algorithms are still restrictive in that they *can not exploit omissive faults.*

This paper introduces a new fault model which is capable to deal with omissive behavior which could be the dominant mode of failure in large distributed systems. It also permits the processes to ignore locally diagnosed errors and hence deal with different number of data items.

Section 2 presents some background material necessary to understand the convergent voting process. Section 3 introduces the new fault model, Omission-MSR (OMSR), and justifies the inclusion of omission faults into the new model. Section 4 presents the convergence rate and fault tolerance for two large families of OMSR voting algorithms which together cover the commonly used voting algorithms. Section 5 compares OMSR and MSR fault-models. It will be shown that the new model provides greater accuracy in fault-tolerance of synchronous voting algorithms. And finally, Section 6 concludes the paper with a summary and some remarks for future research.

2 Background and Definitions

The objective of reaching an approximate agreement is to guarantee that, at the termination of a voting algorithm, the voted value for each non-faulty process is within the range of the initial correct values and that the diameter of the voted values for each of the non-faulty processes is 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. On receipt of a collection of values, each process executes a voting function F, to obtain its latest voted value, which it then broadcasts in the next round of message exchange. The objective of Approximate Agreement is achieved if it is guaranteed that the range of values held by the non-faulty processes is reduced in each round [13, 15, 24, 32, 33]. This property, called single-step convergence, guarantees that the range of values will eventually be less than ϵ, given enough rounds.

In general, there are two forms of voting algorithms: *synchronous* and *asynchronous* [13]. In a synchronous distributed system the processing and the communication delays of non-faulty processes are bounded. There is thus a point in time by which any process executing a convergent voting algorithm will have received all data from all non-faulty processes. Any data arriving after that time is considered to be from a faulty process. By contrast, asynchronous systems impose no bounds on process operation [13]. It is thus impossible to differentiate between a slow non-faulty process and a "dead" faulty process. The analysis of Approximate Agreement in this research is restricted to synchronous systems. These systems are representative of real time functions such as data sensor management and fault-tolerant clock synchronization [19, 24, 26, 29, 31].

2.1 Real–Valued Multisets

Approximate Agreement requires the manipulation of non-disjoint 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. For example, a set of real numbers contains no more than one occurrence of any given value, while a multiset of real numbers may contain multiple occurrences of a value. The number of times a particular object (value) appears in a multiset is called the *Multiplicity* of that object. A finite multiset $\mathbf{V}$ of real values may be represented as a mapping $\mathbf{V} : \Re \rightarrow \aleph$. For each real value r, $V(r)$ is defined as the multiplicity of r in $\mathbf{V}$. The size of $\mathbf{V}$ is $V = |\mathbf{V}| = \sum_{r \in \Re} V(r)$.

An alternative representation for a multiset of real numbers is a monotonically increasing sequence of the real values of its elements, i.e. $\mathbf{V} = \langle v_1, \ldots, v_V \rangle$ ordered such that: $v_i \leq v_{i+1} \; \forall \; i \in \{1, \ldots, V-1\}$ [1, 25]. Both representations of a multiset are equivalent, but for most operations the second notation is more convenient. To avoid confusion, we use upper-case symbols for elements in the real-to-integer mapping form, e.g. $V(r)$. Similarly, we use angle-braces and lower-case symbols for elements in the sequence form, e.g. $\mathbf{V} = \langle v_1, \ldots, v_V \rangle$.

Real-Valued Parameters – A multiset of real numbers has several useful parameters.

$\min(\mathbf{V}) \;=\; \min(r \in \Re : V(r) > 0) \;=\; v_1$; the minimum value of the elements in $\mathbf{V}$.

$\max(\mathbf{V}) \;=\; \max(r \in \Re : V(r) > 0) \;=\; v_V$; the maximum value of the elements in $\mathbf{V}$.

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

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

$\mathrm{mean}(\mathbf{V}) =$ The arithmetic mean of the real values of all elements of $\mathbf{V}$;

$$\mathrm{mean}(\mathbf{V}) \;=\; \frac{1}{V}\left(\sum_{r\in\Re} V(r)\cdot r\right) \;=\; \frac{1}{V}\left(\sum_{i=1}^{V} v_i\right).$$

Multiset Relations – Two multisets $\mathbf{U}$ and $\mathbf{V}$ may be related to each other by Union, Intersection, Sum, or Difference.

Union: Let $\mathbf{W} = \mathbf{V} \cup \mathbf{U}$. Then $W(r) = \max[V(r), U(r)]\ \forall\, r \in \Re$.

Intersection: Let $\mathbf{W} = \mathbf{V} \cap \mathbf{U}$. Then $W(r) = \min[V(r), U(r)]\ \forall\, r \in \Re$.

Sum: Let $\mathbf{W} = \mathbf{V} + \mathbf{U}$. Then $W(r) = V(r) + U(r)\ \forall\, r \in \Re$.

Difference: Let $\mathbf{W} = \mathbf{V} - \mathbf{U}$. Then $\forall\, r \in \Re$:

$$W(r) \;=\; \begin{cases} V(r) - U(r) & \text{if } V(r) > U(r) \\ 0 & \text{otherwise} \end{cases}$$

Subsequences – Intuitively, a subsequence of a multiset is a submultiset whose elements are determined solely by their relative positions in the sequence of the original multiset. For a more formal definition consider two non-empty multisets $\mathbf{V} = \langle v_i \rangle\ \forall\, i \in \{1, \ldots, V\}$ and $\mathbf{U} = \langle u_j \rangle\ \forall\, j \in \{1, \ldots, U\}$, where $\mathbf{U} \subseteq \mathbf{V}$. $\mathbf{U}$ is a subsequence of $\mathbf{V}$ if there is an order-preserving one-to-one mapping k, from the *indices* of $\mathbf{U}$ to the *indices* of $\mathbf{V}$, i.e. $u_j = v_{k(j)}\ \forall\, j \in \{1, \ldots, U\}$ and $k(j) < k(j+1)\ \forall\, j \in \{1, \ldots, U-1\}$.

2.2 Fault Classification

A handful of voting algorithms exist which guarantee convergence. Most of these algorithms assume that *all* faults behave in Byzantine manner. But in many applications true Byzantine faults occur rarely and under complex conditions. This limitation leads to system designs which are more complex and require a greater number of processes than necessary to guarantee convergence. A more realistic approach to designing fault-tolerant distributed systems is to incorporate different types of faults and place a limit on the maximum number of faults in each class. Accordingly, Meyer and Pradhan [27] partitioned the space of all faults into two classes: *Benign* faults and *Malicious* faults. Benign faults are defined as those which are self-incriminating or self-evident to *all* nodes. Malicious faults are the ones which do not qualify as benign faults. Specifically, a malicious fault is caused by a process which behaves maliciously by "lying" when asked for a data value. Thambidurai and Park [30] further partitioned the malicious faults into two subclasses: *Symmetric* (single-valued) faults and *Asymmetric* (Byzantine, active asymmetric, two-faced) faults. A symmetric fault is defined as a fault whose value is perceived identically by all receiving non-faulty processes. An asymmetric fault is the one which is capable of sending conflicting (arbitrary) messages to different non-faulty processes. Given asymmetric faults a, symmetric faults s, and benign faults b, the total number of faults t in the system is then $t = a + s + b$. Thambidurai and Park used this partitioning to derive tighter bounds on the fault-tolerance of Byzantine Agreement algorithms [30].

By employing the Thambidurai and Park model, we have achieved a tighter bound on fault-tolerance and have shown that convergence can be determined more accurately [21, 20]. It will be shown that by partitioning some of the fault modes into disjoint submodes, the fault-tolerance can be further improved. Furthermore, we will show that the OMSR voting algorithms use less time to reach agreement than the existing voting algorithms.

For ease of reference, the single-mode Byzantine fault-model is labeled "BYZ-1". Similarly, we abbreviate the Meyer and Pradhan Hybrid fault model as "MPH-2", and the Thambidurai and Park Hybrid fault model as "TPH-3".

2.3 MSR Voting Algorithms

MSR algorithms employ a particular family of voting algorithms with the general form [15, 20]:

$$F(\mathbf{V}) = \text{mean}\left[Sel_\sigma\left(Red^\tau(\mathbf{V})\right)\right]. \tag{2.1}$$

The "Reduction" function Red^τ removes the τ largest and τ smallest elements from multiset $\mathbf{V}$, i.e. $Red^\tau(\mathbf{V}) = \left\langle v_{(1+\tau)}, \ \dots \ , v_{(V-\tau)} \right\rangle$. The new submultiset is called *medial multiset* $\mathbf{M}$. The "Selection" function Sel_σ then selects a submultiset $\mathbf{S}$ of σ elements from the reduced multiset $Red^\tau(\mathbf{V})$. The final voted value is the arithmetic mean of the selected multiset.

If we restrict the choice of selection functions such that Sel_σ always produces a subsequence of $Red^\tau(\mathbf{V})$, then $F(\mathbf{V})$ is the *Mean* of a *Subsequence* of the *Reduced* multiset. The family of all voting algorithms with this property are called *Mean-Subsequence-Reduced* or MSR algorithms [20]. Members of the MSR family differ from each other only in their definition of the selection function Sel_σ. Some examples of MSR algorithms are the Fault-Tolerant Midpoint and Fault-Tolerant Mean [13], Dolev's Optimal algorithm [15], and the Mixed Mode Optimal, Binary Mean, and Binary Suboptimal algorithms [20].

In the context of MSR algorithms, the following definitions are used frequently:

$\mathbf{P}_i$ = The set of processes adjacent to process i. $\mathbf{P}_j$ is defined similarly for j.

$\mathbf{P}_{i\cap j}$ $= \mathbf{P}_i \cap \mathbf{P}_j$, the set of processes adjacent to both processes i and j.

$\mathbf{P}_{i\cup j}$ $= \mathbf{P}_i \cup \mathbf{P}_j$, the set of processes adjacent to either or both of processes i and j. Completely connected systems have the property that $\mathbf{P}_i = \mathbf{P}_{i\cup j} = \mathbf{P}_{i\cap j}$.

χ $= |\mathbf{P}_i| - |\mathbf{P}_{i\cap j}| = |\mathbf{P}_j| - |\mathbf{P}_{i\cap j}|$, the number of processes whose values are receivable by either i or j, but *not* by both. For completely connected systems, $\chi = 0$.

f = The maximum number of faulty processes in either $\mathbf{P}_i \backslash \mathbf{P}_{i\cap j}$ or $\mathbf{P}_j \backslash \mathbf{P}_{i\cap j}$ regardless of the fault modes exhibited.

$\mathbf{V}_i$ = The multiset of values received in a given round by non-faulty process i. The number of elements in $\mathbf{V}_i$ is $V_i = |\mathbf{V}_i|$.

$\mathbf{M}_i$ $= Red^{\tau}(\mathbf{V}_i)$, the *Medial Multiset* of $\mathbf{V}_i$. The number of elements in $\mathbf{M}_i$ is $M_i = |\mathbf{M}_i| = V_i - 2\tau$.

$\mathbf{S}_i$ $= Sel_{\sigma_i}(Red^{\tau}(\mathbf{V}_i))$, the *Selected Multiset* generated by $F(\mathbf{V}_i)$.

σ_i $= |\mathbf{S}_i|$, the number of elements in $\mathbf{S}_i$.

$\mathbf{U}_{i\cap j}$ = The multiset of *correct* values generated in $\mathbf{P}_{i\cap j}$.

$\mathbf{U}_{i\cup j}$ = The multiset of *correct* values generated in $\mathbf{P}_{i\cup j}$.

MSR algorithms assume that the multisets $\mathbf{V}$ for non-faulty processes are all of the same size. Consequently, the sizes of medial multisets and σ for all not-faulty processes will be the same as well. Therefore, when there is no room for ambiguity, subscripts may be omitted.

2.4 Single-Step Convergence

It is known that Approximate Agreement can be achieved if a voting algorithm is *single-step convergent.* However, the precise definition of single-step convergence depends on whether the system is completely connected or partially connected.

2.4.1 Completely Connected Systems

In a completely connected system, $\mathbf{U}_i \equiv \mathbf{U}_{i\cap j} \equiv \mathbf{U}_{i\cup j}$. Each non-faulty process i executes a convergent voting algorithm, producing voted value $F(\mathbf{V}_i)$. A voting algorithm is single-step convergent if both of the following conditions are true for every voting round:

[C1] For each non-faulty process i, $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{i\cup j})$.

[C2] For each pair of non-faulty processes (i,j), $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq C\,\delta(\mathbf{U}_{i\cup j})$, where $0 \leq C < 1$.

2.4.2 Local Convergence with Partial Connectivity

A partially connected system differs from a completely connected system in that a given process i does not receive values from all non-faulty processes. Rather, it receives values only from processes in $\mathbf{P}_i$. There are now two types of convergence to be considered: local convergence over a specified subgraph, and global convergence over the entire system graph. It has been shown that local convergence is necessary for global convergence [21]. It has also been shown that in a fault-free system, local agreement is both necessary and sufficient to ensure eventual global agreement [21]. However, the fault-tolerance and convergence rates to achieve global convergence are open questions at this time. Global convergence is thus the topic of ongoing research and is beyond the scope of this paper.

This paper addresses local convergence without the benefit of message relays. Several constraints are placed on the system: (1) The system topology is a non-hierarchical, *regular, homogeneous*, undirected graph of N processing nodes, each with degree d, and (2) messages may not be relayed between processes; thus, each non-faulty process i receives *only* the values of its immediate neighbors (including itself). Hence, $V_i \leq |\mathbf{P}_i| = d+1$, (3) $N \gg d$ so that "wrap-around" effects can not assist the convergence process.

In a partially connected system, $\mathbf{U}_{i\cap j} \subset \mathbf{U}_{i\cup j}$, yielding two distinct criteria for local convergence.

Intersection Convergence: Given a voting algorithm $F(\mathbf{V})$, two processes i and j are single-step convergent if the following conditions are both true:

[I1] $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{i\cap j})$, and $F(\mathbf{V}_j) \in \rho(\mathbf{U}_{i\cap j})$,

[I2] $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq C\,\delta(\mathbf{U}_{i\cap j})$, where $0 \leq C < 1$.

Union Convergence: Given a voting algorithm $F(\mathbf{V})$, two processes i and j are single-step convergent if the following conditions are both true:

[U1] $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{i\cup j})$, and $F(\mathbf{V}_j) \in \rho(\mathbf{U}_{i\cup j})$,

[U2] $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq C\,\delta(\mathbf{U}_{i\cup j})$, where $0 \leq C < 1$.

2.5 Hybrid Analysis of MSR Algorithms

To date, MSR voting algorithms are the only family of convergent voting algorithms to have been analyzed under a hybrid fault model. Results have shown that the fault-tolerance of these algorithms is significantly better than predicted by the single-mode Byzantine fault model. In addition, a simple relation has been derived for the convergence rate of any MSR algorithm [20, 21].

2.5.1 Fault-Tolerance

MSR algorithms assume that the voting multisets for two arbitrary non-faulty processes i and j are of the same size. Let this size be denoted by $V_\circ$. Then to ensure that $V_\circ = V_i = V_j$, if less than $V_\circ$ values are received, an arbitrary default value is chosen for each value not received.

A major difference between completely connected and partially connected systems is their handling of benign faults. In a completely connected system, benign faults can be ignored because *all* processes can delete the benign errors from their voting multisets and vote with a smaller size multiset [20], so that $V_\circ = V_i - b$. However, in a partially connected system without message relays, no value is received by all processes. Thus, *no fault is self-evident to all non-faulty processes* as required in the definition of a benign fault [27]. Therefore, in partially connected systems, only symmetric and asymmetric faults are considered, so that $V_\circ = V_i = V_j$.

Although malicious errors play an important role in determining the convergence rate and fault tolerance expressions, the results revealed that it is the *effective* number of asymmetric values, which includes both erroneous and non-erroneous values, that is actually the basis for these expressions. Completely connected systems have the property that $\mathbf{P}_i = \mathbf{P}_j = \mathbf{P}_{i \cap j}$. Therefore, the effective number of asymmetric values between two non-faulty processes is the number of faulty processes which send conflicting messages to i and j, i.e. the number of asymmetric faults a. However, in partially connected systems, $\mathbf{P}_i \neq \mathbf{P}_j$ and thus a non-faulty process $k \in \mathbf{P}_i \backslash \mathbf{P}_{i \cap j}$ communicates with process i but not with process j. Similarly, a process $\ell \in \mathbf{P}_j \backslash \mathbf{P}_{i \cap j}$ communicates with j but not with process i. In the worst case, two processes k and ℓ can send different values to processes i and j, respectively. Thus, each process pair (k, ℓ) can have the same impact on $\mathbf{V}_i$ and $\mathbf{V}_j$ as a single asymmetrically faulty process in $\mathbf{P}_{i \cap j}$. This effect can occur regardless of the fault status of k and ℓ. The number of such process pairs is χ. Accordingly, in the worst case, the effective number of asymmetric values is $(a + \chi)$.

To show the conditions under which MSR algorithms are convergent, the following parameters are introduced:

α	$=$	The *effective* number of asymmetric values in a voting multiset $\mathbf{V}$.
β	$=$	The *fault burden* of the system, i.e. the minimum number of processes in $\mathbf{P}_{i \cap j}$ to guarantee convergence.
$\tau_\circ$	$=$	The *effective* number of malicious errors in a voting multiset $\mathbf{V}$, i.e. the number of malicious processes in $\mathbf{P}_{i \cap j}$ plus the number of processes in $\mathbf{P}_i \backslash \mathbf{P}_{i \cap j} = \mathbf{P}_j \backslash \mathbf{P}_{i \cap j}$ that must be treated as malicious.

An MSR algorithm is convergent if the following are all true [20]:

$$\tau \geq \tau_\circ, \tag{2.2}$$

$$V_\circ \geq 2\tau + \max(\,\alpha + 1, \sigma\,), \tag{2.3}$$

$$|\mathbf{P}_{i \cap j}| \geq \beta. \tag{2.4}$$

where the values for $\tau_\circ$, α, and β are defined in the following table:

Connectivity	$\tau_\circ$	α	β
Complete	$a+s$	a	$V_\circ + b$
Union	$a+s+f$	$\chi + a$	$V_\circ - \chi$
Intersection	$a+s+\chi$	$\chi + a$	$V_\circ - \chi$

Table 1: Summary of MSR Convergence Parameters

2.5.2 Convergence Rate

The convergence rate of an MSR algorithm depends on two parameters of the selection function $Sel_\sigma(\mathbf{M})$ [20]. The first parameter, σ, is the size of the selected multiset $\mathbf{S}$. The second parameter, γ, is a measure of how uniformly the elements of $\mathbf{S}$ are distributed within the medial multiset $\mathbf{M}$, which is defined below[3].

Recall that selected multiset $\mathbf{S} = \langle s_1, \dots, s_\sigma \rangle$ is a *subsequence* of medial multiset $\mathbf{M} = \langle m_1, \dots, m_M \rangle$. Let g be an index into $\mathbf{S}$. Then, for each $g \in \{1, \dots, \sigma\}$ there exists exactly one $k(g) \in \{1, \dots, M\}$ which guarantees that $s_g = m_{k(g)}$ *for all possible* $\mathbf{M}$. Given the multisets $\mathbf{M}_i$ and $\mathbf{M}_j$, let $\mathbf{M}_\ell$ be any multiset of size $\min(M_i, M_j)$. Then, given subsequence $\mathbf{S}_\ell = Sel_{\sigma_\ell}(\mathbf{M}_\ell)$ and two indices g,$h \in \{1, \dots, \sigma_\ell\}$ such that $g \leq h$, we define $\Delta k(g,h)$ as the number of elements in the submultiset $\left\langle m_{k(g)+1}, \dots, m_{k(h)} \right\rangle$ of $\mathbf{M}_\ell$ spanned by elements $\langle s_g, \dots, s_h \rangle$ in $\mathbf{S}_\ell$. So, $\Delta k(g,h) = [k(h) - k(g)]$.

For a given non-negative integer $\alpha < M_\ell$, it will be useful to know the minimum value of $(h-g)$ for which $\Delta k(g,h) \geq \alpha$. Thus, for each $g \in \{1, \dots, \sigma_\ell\}$, define $\Delta(g)$ as follows:

IF: $\Delta k(g, \sigma_\ell) \geq \alpha$,
THEN: $\Delta(g) =$ the minimum value of the difference $(h-g)$ such that $\Delta k(g,h) \geq \alpha$,
ELSE: $\Delta(g)$ does not exist for this value of g.

[3]Herein, the original definition of γ defined for MSR algorithms has been modified to ensure that MSR and OMSR algorithms can uniformly reference the same definition

Finally, γ_α is defined as:

$$\gamma_\alpha \; = \max_{\forall\; g \in \{1, \ldots, \sigma_\ell\}} (\Delta(g)). \tag{2.5}$$

Accordingly, given any index g into $\mathbf{S}$, for which $(g+\gamma) \leq \sigma$, it is assured that $\Delta k(g, g+\gamma) \geq \alpha$. In other words, the submultiset $\langle s_g, \ldots, s_{g+\gamma_\alpha} \rangle$ is guaranteed to span α elements of $\mathbf{M}_\ell$, for any $g \in \{1, \ldots, \sigma_\ell - \gamma_\alpha\}$.

As a practical matter, obtaining the value of γ_α is simple, given constants M_ℓ, α, and a specified selection function Sel_{σ_ℓ}. For each $g \in \{1, \ldots, \sigma_\ell\}$, $\Delta(g)$ is found by inspection. Then, γ_α is simply the maximum of all existing $\Delta(g)$ values. If there is no value of g for which $\Delta(g)$ exists, then γ_α does not exist.

The Fault-Tolerant Mean algorithm provides a simple example [13]. This algorithm selects *all* elements of medial multisets $\mathbf{M}_i$ and $\mathbf{M}_j$ for inclusion in $\mathbf{S}_i$ and $\mathbf{S}_j$ respectively. Accordingly, $\mathbf{S}_\ell = \mathbf{M}_\ell$, $\sigma_\ell = M_\ell$, and $s_g = m_g \;\forall\; g \in \{1, \ldots, \sigma_\ell\}$. It follows that $\Delta k(g, g+1) = 1 \;\forall\; g \in \{1, \ldots, \sigma_\ell - 1\}$. Thus, $\Delta k(g, g+\alpha) = \alpha \;\forall\; g \in \{1, \ldots, \sigma_\ell - \alpha\}$, which implies that $\Delta(g) = \alpha \;\forall\; g \in \{1, \ldots, \sigma_\ell - \alpha\}$. Therefore, $\gamma_\alpha = \alpha$.

The significance of γ_α is in its importance to the convergence rate. It has been shown [20] that the convergence rate C of any synchronous MSR algorithm is given by the simple expression:

$$C \;=\; \frac{\gamma_\alpha}{\sigma}.$$

3 Omissive/Transmissive Fault Model

Analysis of MSR voting algorithms under the TPH-3 model has revealed that under the most likely fault scenarios, fault-tolerance is significantly better than predicted by the BYZ-1 model. However, there is still significant room for improvement based on further partitioning of fault modes. What follows is a continuation of this work, which further partitions the asymmetric and symmetric fault modes of the TPH-3 model into *transmissive* and *omissive* modes.

We begin by partitioning each round of an MSR voting algorithm into two sequential phases: a *sampling* phase, and an *execution* phase. During the sampling phase, each process i collects incoming values, filters out the recognized erroneous values, substitutes default values for missing values, and produces the voting multiset $\mathbf{V}_i$. During the execution phase, process i executes its voting function to reduce $\mathbf{V}_i$ to a single voted value $F(\mathbf{V}_i)$.

3.1 Fault Model Definitions

As used herein, a transmissive fault is one which delivers erroneous value(s) to one or more receiving processes. By contrast, an omissive fault fails to deliver any value to one or more receiving processes. Thus, a transmissive fault delivers a *transmissive error* while an omissive fault delivers an *omissive error*. The precise definition of transmissive and omissive behavior differs among the three fault modes of the TPH-3 model.

1. *Benign* errors are deleted from $\mathbf{V}_i$ by the sampling phase before it is passed to the execution phase. Thus, as viewed by the execution phase, all benign faults are by default omissive.

2. By definition, a *symmetric* fault delivers the same value (or absence of value) to all receivers. Thus, a symmetric fault may be either transmissive or omissive, but not both.

3. An *asymmetric* fault can be simultaneously transmissive and omissive with respect to different receiving processes. For example, a faulty process can deliver one value to one receiver, a second value to a second receiver, and no value to a third receiver. We must therefore be more precise in differentiating between transmissive and omissive behavior in asymmetric faults. Specifically:

    (a) *Omissive Asymmetry* is the case where a faulty process sends a single value to a subset of receivers, and no value to all remaining receivers,

    (b) *Transmissive Asymmetry* is the case where a faulty process exhibits any form of asymmetric behavior other than omissive asymmetry.

We define a subclass of omissive asymmetric faults as *strictly omissive* in which a process sends a single "correct" value to some processes and no value to other processes. Under this new hybrid model, the total number of faulty processes in the system is:

$$t = (a' + \omega_{a1}) + (s' + \omega_s) + b, \tag{3.1}$$

where:

ω_{a1}	= The number of strictly omissive asymmetric faults in the system,
a'	= The number of faults in the system displaying any form of asymmetric behavior other than strictly omissive,
s'	= The number of transmissive symmetric faults in the system,
ω_s	= The number of omissive symmetric faults in the system,
b	= The number of benign faults in the system.

We will abbreviate this five-mode Omissive/Transmissive Hybrid fault model as "OTH-5". The relationships between the OTH-5 and TPH-3 model are specified by the relations: $a = (a' + \omega_{a1})$ and $s = (s' + \omega_s)$. Figure 1 illustrates the relationships among all fault models discussed herein.

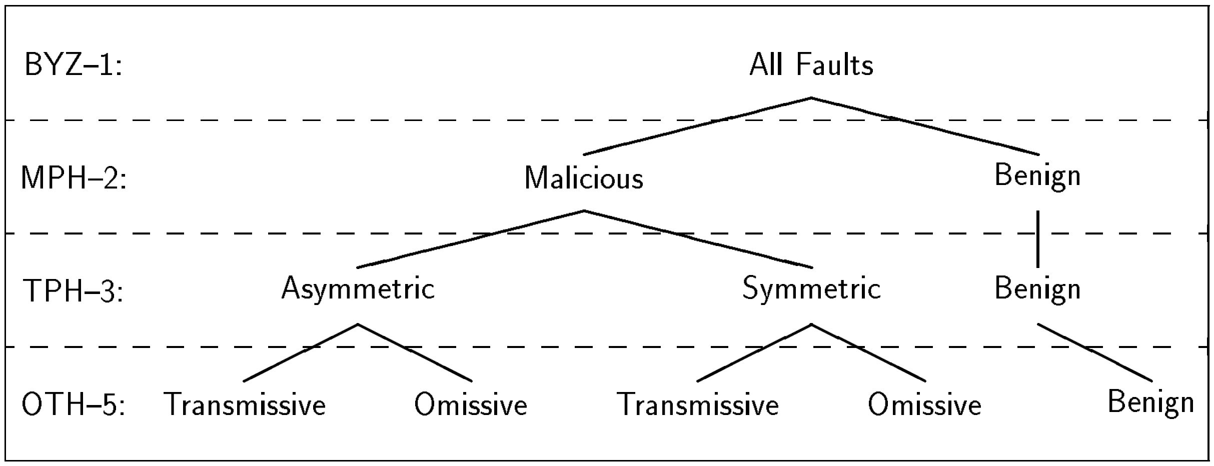


Figure 1: Relationships Between Fault Models

3.2 Model Justification

Any number of hypothetical fault models can be postulated and studied. However, the practical utility of a fault model depends on whether its fault modes can be traced back to specific physical faults in a system. Thambidurai and Park noted that in a fault-tolerant distributed system, each "process" occupies a separate processor or computer. Furthermore, each source process can be partitioned into two modules, which we call the *generator module* and the *delivery module*.

1. The generator module comprises those components responsible for correctly generating a value v. It generally includes the processor, memory system, internal bussing, and all input devices.

2. The delivery module comprises those components responsible for correctly transmitting the generated value v to all receivers. It generally includes a hardware-based transmitter, bus drivers, and the transmission medium.

The value of this partitioning lies in the potential behaviors of the two modules. If the delivery module is non-faulty, then any error caused by a faulty generator module *will be delivered symmetrically* by the delivery module. Thus, an asymmetric fault must originate within the delivery module. In general, the generating module contains significantly more hardware and software than the delivery module. Indeed, the system can be designed such that each processor's delivery module contain only a few simple integrated circuit chips [19]. Hence, the vast majority of faults are expected to occur within the generator module, and are thus guaranteed to be symmetric.

Similar observations can be made regarding omissive and transmissive behavior. For example, a process containing an omissive symmetric fault simply fails to transmit any messages. This behavior is equivalent to the common definition of a *crash* fault or a *fail-stop* fault [10, 11]. Any fault which causes the generating module of a process to halt or "hang" is then an omissive symmetric fault. In many systems, such faults are considered extremely likely [5, 8, 9]. In addition, even a modest amount of internal self-checking can dramatically increase the probability that a fault will behave omissively [34, 35]. Finally, some delivery modules such as Boeing's DATAC bus terminal [16] have been designed with significant internal self-checking to strongly bias its fault behavior toward the fail-stop mode. It is then reasonable to assume that a significant fraction of symmetric faults will be omissive.

A strictly omissive asymmetric fault corresponds to the definition of "omission fault" used by several other authors [6, 7, 10]. Its behavior also corresponds to the rather common assumption that messages are protected by unforgeable "authenticators" [14, 28]. In reality, this form of asymmetry can readily occur in systems where messages are protected by any form of data redundancy (e.g. Error Correction Codes or Cyclic Redundancy Checks) [17].

Any form of asymmetry other than strictly omissive requires the delivery of an *undetectably incorrect* value to one or more receivers. This in turn requires either that the generating module delivers an incorrect value to the faulty delivery module (a double fault within one process) or that the delivery module alters the original value in such a way that it "escapes" detection by the data redundancy checks of the receiver. Since both of these events are relatively unlikely, strictly omissive asymmetric faults could easily constitute the vast majority of all asymmetric faults.

4 OMSR Voting Algorithms

In all previous studies of synchronous Approximate Agreement, it was required that the size of the voting multiset received by the execution phase of the voting algorithm be identical for all processes, i.e. $|\mathbf{V}_i| = |\mathbf{V}_j| \ \forall\ i, j \in \{1, \ \dots\ , N\}$. Thus, if an omissive error occurred, the sampling phase was required to substitute a default value for the missing value. This action had the effect of *transforming omissive errors into transmissive errors*. The only exception allowed was for a value transmitted by a globally recognized faulty process (i.e. a benign fault). This value could be discarded by the sampling phase because it was assured *a priori* that all other non-faulty sampling phases would do likewise, preserving the equality $|\mathbf{V}_i| = |\mathbf{V}_j|$. The net result is that *none of the previous voting algorithms can exploit the* *omissive behavior of malicious faults.*

Similarly, there was no ability to simply disregard locally diagnosed transmissive errors, such as parity errors. The sampling phase either had to use the erroneous value as received, or substitute the default value into $\mathbf{V}$. The net result is that *none of the previous voting algorithms can exploit local diagnosis of self-evident errors.*

This paper presents a variant of the MSR family of voting algorithms which can exploit omissive behavior. The new family of algorithms, called *Omission-MSR*, or OMSR algorithms, differs from MSR algorithms only in the sampling phase. Specifically, when an OMSR algorithm receives either an omissive error or a self-evident transmissive error, it simply discards that value. No defaults are substituted into the voting multiset $\mathbf{V}$. Thus, omissive errors remain omissive, and self-evident errors become omissive. As a result, the equality of $|\mathbf{V}_i|$ and $|\mathbf{V}_j|$ is no longer preserved.

4.1 Definitions and Notation

Recall that ω_{a1} is the number of strictly omissive asymmetric faults in the system. Assume that ω_{a1_i} is the number of processes whose values are received by process i but not by process j. Similarly, assume ω_{a1_j} is the number of processes whose values are received by process j but not by process i. However, it is possible that a strictly omissive asymmetric process may send values to processes other than i and j. Therefore, $\omega_{a1_i} + \omega_{a1_j} \leq \omega_{a1}$.

Recall that OMSR algorithms are different from each other only in their definition of the selection function. It is the selection function that determines the values of σ_i and σ_j. There are thus two general families of selection functions: *enumerative* and *non-enumerative*. In an enumerative selection function, the relative position of each selected element in $\mathbf{M}$ *does not* depend on M. More formally, for two different size medial multisets $\mathbf{M}_i$ and $\mathbf{M}_j$, if $s_{i,g} = m_{i,k}$, then $s_{j,g} = m_{j,k}$, $\forall\ k \in \{1, \ldots, \min(M_i, M_j)\}$. Whereas, in non-enumerative selection functions, the position of *at least* one of the elements selected must depend on M.

There are some known selection functions which are enumerative, such as Fault-Tolerant

Mean, Dolev's Optimal algorithm [15], and Mixed-Mode Optimal algorithm [20]. However, there are voting algorithms which are not enumerative. For instance, Fault-Tolerant Midpoint [13] selects only the two extremal values of a medial multiset, i.e. m_1 and m_M. Due to the existence of omission faults, two arbitrary medial multisets $\mathbf{M}_i$ and $\mathbf{M}_j$ could be of different sizes, so that the relative positional value of the right extremal value of $\mathbf{M}_i$ may not be the same as that of $\mathbf{M}_j$, i.e. $M_i \neq M_j$.

4.2 Dynamic-σ Selection Functions

There could be many sub-families of enumerative selection functions. The sub-family adopted for our fault model is the *dynamic-σ* selection functions defined below.

Let us define the medial multiset $\mathbf{M}_{max} = \langle m_1, \ldots, m_{M_{max}} \rangle$, where M_{max} is the maximum possible size of all medial multisets in the system. Also, let us define the enumerative selection set as a set of integers $E = \{e_1, \ldots, e_{\sigma_{max}}\}$, where $e_j < e_{j+1} \; \forall \; j \in \{1, \ldots, \sigma_{max} - 1\}$ and σ_{max} is the number of elements selected from $\mathbf{M}_{max}$, such that $\mathbf{S}_{max} = Sel_{\sigma_{max}}(\mathbf{M}_{max}) = \left\langle m_{e_1}, m_{e_2}, \ldots, m_{e_{\sigma_{max}}} \right\rangle$. In other words, E lists the indices of all elements of $\mathbf{M}_{max}$ selected for inclusion in $\mathbf{S}_{max}$. Then, for an arbitrary $\mathbf{V}_i$, let the selected multiset $\mathbf{S}_i$ be $Sel_{\sigma_i}(\mathbf{M}_i) = \left\langle m_{e_1}, \ldots, m_{e_{\sigma_i}} \right\rangle$, where σ_i is the largest value such that $e_{\sigma_i} \leq |\mathbf{M}_i|$. In other words, $\mathbf{S}_i$ includes the elements from $\mathbf{M}_i$ whose indices are listed in E and are less than or equal to $|\mathbf{M}_i|$.

The number of elements selected depends on the size of $\mathbf{M}_i$ which is:

$$M_i = V_i - 2\tau \tag{4.1}$$

Let $\mathbf{V}_\circ$ represent any voting multiset with the largest number of data values received. Furthermore, assume $\mathbf{U}_\circ$ to be the multiset with the smallest number of *correct* values allowed in the system. Then, due to the definition of dynamic-σ selection functions, the number of elements selected by process i is maximized when $M_i = M_{max}$. That occurs, for instance in a fault-free system, when $\mathbf{V}_i$ receives a value from every process, i.e. $V_i = V_\circ = (U_\circ + a' + \omega_{a1} + s' + \omega_s)$. Similarly, the smallest medial multiset $\mathbf{M}_{min}$ is obtained when $\mathbf{V}_i$ contains the minimum number of correct elements, i.e. $U_\circ$, plus those faults which can not behave omissively, i.e. s'. Therefore, $V_i = U_\circ + s'$. Accordingly, by substituting the minimum and maximum values of V_i into (4.1), we obtain the following:

$$\begin{aligned} M_{max} &= V_\circ - 2\tau && (4.2) \\ M_{min} &= (U_\circ + s') - 2\tau && (4.3) \end{aligned}$$

Once M_{max} and M_{min} are determined, the enumeration set E can be applied to obtain σ_{max} and σ_{min}, respectively.

Using (4.1), it is useful to observe the following:

$$\begin{aligned} M_{max} - M_{min} &= V_\circ - (U_\circ + s') \\ &= [U_\circ + (a' + \omega_{a1} + s' + \omega_s)] - (U_\circ + s') \\ &= a' + \omega_{a1} + \omega_s && (4.4) \end{aligned}$$

$$V_i \geq V_j \;\Leftrightarrow\; M_i \geq M_j \;\Leftrightarrow\; \sigma_i \geq \sigma_j \qquad (4.5)$$

4.3 Fixed-σ Selection Functions

Recall that a selection function is a mapping from the set $\{1, \ldots, \sigma\}$ to the codomain set $\{k(1), k(2), \ldots, k(\sigma)\}$. A selection function is non-enumerative if at least one of its codomain set values is a function of M, i.e. the size of the medial multiset.

Like enumerative selection functions, there exist many sub-families of non-enumerative functions. We adopt the name *fixed-σ* for one such sub-family, where $\sigma_i = \sigma_j$, regardless of the sizes of $\mathbf{M}_i$ and $\mathbf{M}_j$. That makes the convergence rate expression simpler because $\sigma_{max} = \sigma_{min}$. However, for two arbitrary processes i and j, if $M_i \neq M_j$, then the sets of codomain values obtained for processes i and j will not be the same. Specifically, the value $k(g)$, $g \in \{1, \ldots, \sigma\}$, for process i may not be the same as that for process j. This problem does not exist with dynamic-σ selection functions. Therefore, to distinguish between $k(g)$ of different processes, the following are defined:

$k_i(g)$ = Is the same as $k(g)$ defined before, except that $k(g)$ is associated with $\mathbf{M}_i$.

$\Delta k_i(g,h)$ = $[k_i(h) - k_i(g)]$, is the number of elements of $\mathbf{M}_i$ in $\left\langle m_{k_i(g)+1}, \ldots, m_{k_i(h)} \right\rangle$.

Other than $\sigma_i = \sigma_j$, fixed-σ selection functions have the following properties:

$$M_i \geq M_j \Rightarrow \begin{cases} k_i(g) \geq k_j(g), & \forall\, g \in \{1, \ldots, \sigma\} \\ \Delta k_i(g, g+1) \geq \Delta k_j(g, g+1), & \forall\, g \in \{1, \ldots, \sigma - 1\} \end{cases}$$

Informally, these properties state that as M increases the number of elements between each pair of selected elements and the index of any selected element in $\mathbf{M}$ increase.

4.4 Convergence with OMSR Algorithms

As indicated previously, OMSR algorithms do not require processes to deal with a fixed number of data items. Therefore, the number of selected elements for two processes i and j could be different. This affects the convergence rate. Accordingly, let us define the following:

$$\mu = \frac{\sigma_{max} - \sigma_{min}}{\sigma_{max}}$$

This is the fraction that "offsets" the convergence rate of OMSR algorithms in comparison to MSR algorithms [3, 4]. This fraction does not exist in MSR algorithms because $\sigma_{max} = \sigma_{min}$ and so $\mu = 0$.

An OMSR algorithm can be convergent only if [3, 4]:

$$\tau \geq \tau_{\circ}, \tag{4.6}$$

$$V \geq 2\tau + \max(\alpha + 1, \sigma_{min}), \tag{4.7}$$

$$|\mathbf{P}_{i \cap j}| \geq \beta. \tag{4.8}$$

It is seen that these equations are very similar to equations (2.2) – (2.4) except that σ is replaced with σ_{min}. Based on these equations, Table 2 shows the values for $\tau_{\circ}$, α, and β.

Connectivity	$\tau_\circ$	α	β
Complete	$a' + s'$	$a' + \omega_{a1}$	$V_\circ + b$
Union	$a' + s' + f$	$\chi + a' + \omega_{a1}$	$V_\circ - \chi$
Intersection	$a' + s' + \chi$	$\chi + a' + \omega_{a1}$	$V_\circ - \chi$

Table 2: Summary of OMSR Convergence Parameters

The convergence rate of an OMSR voting algorithm is:

$$C = \mu + \frac{\gamma_\alpha}{\sigma_{max}}.$$

Recall that OMSR algorithms do not require equality in medial multiset sizes. However, fixed-σ voting algorithms require equality in the number of selected elements, i.e. $\sigma_i = \sigma_j$, so that $\sigma_{max} = \sigma_{min}$. Therefore, $\mu = 0$ for fixed-σ voting algorithms.

4.5 Application Example

Previous section indicated that convergence can be guaranteed if:

$$V_\circ \;\geq\; 2\tau + \max(\alpha + 1, \sigma_{min}) \tag{4.9}$$

To determine the minimum value of $V_\circ$ for which a convergent OMSR voting algorithm exists, consider the lower bounds on σ_{min} when: (1) $\alpha = 0$, (2) $\alpha > 0$.

Case 1 : If the number of effective asymmetric values α is zero, i.e. the system is fault free, then $\gamma_\alpha = 0$. To guarantee convergence, it is only necessary that $\mathbf{S}_{min}$ be non-empty, so any $\sigma_{min} \geq 1$ will suffice. Substituting the lower bound of σ_{min} into (4.9) then yields:

$$V_\circ \;\geq\; 2\tau + \alpha + 1 \tag{4.10}$$

Case 2 : If $\alpha > 0$, then σ_{min} must be at least two, to ensure that γ_α exists. Substitution of $\sigma_{min} = 2$ into (4.9) then produces:

$$V_\circ \;\geq\; 2\tau + \alpha + 1 \tag{4.11}$$

By noting that in both cases, (4.10) and (4.11), $\max(\alpha+1, \sigma_{min}) = (\alpha+1)$, shows that a convergent OMSR voting algorithm exists if:

$$V_\circ \geq 2\tau + \alpha + 1 \tag{4.12}$$

Let V_U and V_I represent $V_\circ$ of Union and Intersection convergence respectively. Then, from Table 2, by substituting for τ and α in (4.12), the lower bound on V_U and V_I become:

$$\begin{aligned} V_U &\geq (3a' + 2s' + 1) + (\chi + 2f) + \omega_{a1} \\ V_I &\geq (3a' + 2s' + 1) + 3\chi + \omega_{a1} \end{aligned}$$

Three common meshes are shown in Figure 2, with degrees $d = 4$, $d = 6$, and $d = 8$. Since each node receives its own value, then $V \leq d + 1$. For each network, two nodes are chosen and labeled i and j such that $|\mathbf{P}_{i \cap j}|$ is maximized. In this figure, the nodes enclosed within a dashed box comprise $\mathbf{P}_{i \cap j}$ for each network.

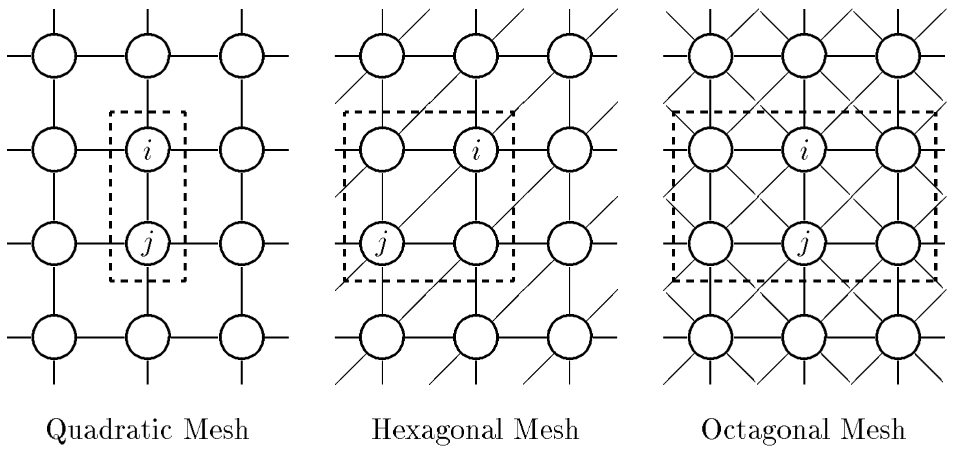


Figure 2: Common Mesh Networks

For readability sake in the following, let us use V_Q, V_H, V_O to represent $V_\circ$ of quadratic, hexagonal, and octagonal meshes respectively.

The inspection of the figure shows that in all three meshes $\chi = 3$ for nodes i and j. Thus, in a fault-free case, Intersection-Convergence yields:

$$\begin{aligned} 5 = d + 1 &\geq V_Q = V_I \geq (3a' + 2s' + 1) + 3\chi + \omega_{a1} = 10 \\ 7 = d + 1 &\geq V_H = V_I \geq (3a' + 2s' + 1) + 3\chi + \omega_{a1} = 10 \\ 9 = d + 1 &\geq V_O = V_I \geq (3a' + 2s' + 1) + 3\chi + \omega_{a1} = 10 \end{aligned}$$

Since all these relations are false, no convergent OMSR algorithm exists for any of these meshes.

In a fault-free case, Union-Convergence gives:

$$\begin{aligned} 5 = d+1 &\geq V_Q = V_U \geq (3a'+2s'+1)+(\chi+2f)+\omega_{a1} = 4 \\ 7 = d+1 &\geq V_H = V_U \geq (3a'+2s'+1)+(\chi+2f)+\omega_{a1} = 4 \\ 9 = d+1 &\geq V_O = V_U \geq (3a'+2s'+1)+(\chi+2f)+\omega_{a1} = 4 \end{aligned} \tag{4.13}$$

Thus, all three meshes are Union-Convergent in the fault-free case. It can be seen that, in any single-fault scenario, both the Hexagonal and Octagonal meshes are Union-Convergent, whereas the Quadratic mesh can handle only a single omissive fault. Furthermore, the Octagonal mesh can tolerate a double-fault if $a' \leq 1$, whereas if omissive faults are not coexistent, non of these networks can tolerate a double asymmetric fault in $\mathbf{P}_{i \cap j}$.

This example reveals that a network is more robust: (1) if Union-Convergence is used because it requires less restrictive conditions on convergence rate than Intersection-Convergence, (2) if the network is more densely connected, (3) if a good fraction of faults are strictly omissive. Strictly omissive faults could be a dominant mode of failure in large geographically distributed networks due to lost or delayed messages.

5 Comparison with MSR Voting Algorithms

Convergence Rate – Recall that the convergence rate of OMSR voting algorithms is:

$$C_{OMSR} = \mu + \frac{\gamma_\alpha}{\sigma_{max}} \tag{5.1}$$

and that of MSR voting algorithms is:

$$C_{MSR} = \frac{\gamma_\alpha}{\sigma} \tag{5.2}$$

Since $a = (a' + \omega_{a1})$, then the values of α for both MSR and OMSR models are the same. In addition, the number of selected elements in MSR is always fixed, so that $\sigma = \sigma_{min} = \sigma_{max}$.

Therefor, (5.2) can be written as:

$$C_{MSR} = \frac{\gamma_\alpha}{\sigma_{max}} \tag{5.3}$$

In general, since $C_{OMSR} \geq C_{MSR}$, the OMSR rate of convergence is worse than that of MSR voting algorithms. To improve C_{OMSR}, the following options can be employed:

1. Use fixed-σ voting algorithms,
2. Use dynamic-σ but ensure that $e_i \leq M_{min}\ \forall\ e_i \in E$.

In either case, since $\sigma_{max} = \sigma_{min}$, $\mu = 0$ which implies that $C_{OMSR} = C_{MSR}$.

Fault-Tolerance – OMSR algorithms differ from MSR algorithms only in the sampling phase. Specifically, when an OMSR algorithm receives either an omissive or a self-evident error, it simply discards that value. The advantage of this approach can be illustrated by comparing the fault burden of MSR model under completely connected systems with that of OMSR model. Equation (4.12) showed that:

$$V_\circ \ \geq\ 2\tau + \alpha + 1 \tag{5.4}$$

Under the MSR model, using Table 1 and relations (2.4) and (5.4), the fault burden is:

$$\begin{aligned} |\mathbf{P}_{i\cap j}| &\geq 2\tau + \alpha + b + 1 \\ &= 2(a+s) + a + b + 1 \\ &= 3a + 2s + b + 1 \\ &= 3(a' + \omega_{a1}) + 2(s' + \omega_s) + b + 1 \\ &= (3a' + 2s' + b + 1) + 3\omega_{a1} + 2\omega_s \end{aligned} \tag{5.5}$$

Similarly, using Table 2 and relations (4.8) and (5.4), the fault burden for OMSR model is:

$$\begin{aligned} |\mathbf{P}_{i\cap j}| &\geq 2\tau + \alpha + b + 1 \\ &= 2(a' + s') + a' + \omega_{a1} + b + 1 \\ &= (3a' + 2s' + b + 1) + \omega_{a1} \end{aligned} \tag{5.6}$$

Thus the value of $|\mathbf{P}_{i \cap j}|$ predicted by (5.6) is $2(\omega_{a1} + \omega_s)$ processes less than (5.5). By a similar observation, partially connected systems under OMSR model show an improvement of $2(\omega_{a1} + \omega_s)$ processes. The reason for the improvement is that MSR algorithms must deal with a fixed number of data items in each round of voting. So, omissions are either presumed not to occur, or the voting algorithm employs defaults values for those values not received. The later has the effect of transforming the omissive behavior into a more severe fault mode such as asymmetric or symmetric. Whereas, an OMSR process, upon the detection of an error, can simply drop the erroneous value from its voting multiset $\mathbf{V}$ and not use it in the execution phase. So, omissive faults are treated as omissive.

In addition, under OMSR, if the error is detected by every process, the effect will be the same as a globally diagnosed benign error. But, if the error is detected by just a subset of processes, then diagnosis at the global level will be of no help. So, in either case, employing OMSR algorithms reduces the duration of the sampling phase. Finally, as observed in Subsection 3.2, the majority of malicious faults can be omissive. Thus, they can be the dominant mode of failure. As a result, the amount of improvement just shown can be very significant in designing fault tolerant systems.

6 Summary and Conclusions

The problem of reaching Approximate Agreement in the simultaneous presence of five types of fault modes (asymmetric omission, symmetric omission, asymmetric, symmetric, and benign) has been addressed. The objectives were: (1) To adopt a fault model capable of masking the effects of different types of fault modes. The fault model is complete in the sense that it can degenerate to lesser number fault models, by simply setting the appropriate parameters (ω_{a1}, ω_s, a, s, b) to zero. (2) To develop a new class of convergent voting algorithms (OMSR) capable of exploiting omission faults. (3) To achieve simple expressions for these algorithms, so that convergence rate and fault-tolerance can be determined easily.

A distinctive feature of OMSR, in comparison to previous fault models on Approximate

Agreement [21, 20, 27, 30, 33], is the integration of omissive faults. Consequently, each process may deal with a different size voting multiset. In contrast, previous models *could not* exploit the existence of omissive faults, because of the assumption that each process must operate on the same size multiset. To avoid dealing with different size multisets, these models substituted default values for omissions. As a result, omissions were transferred into symmetric or asymmetric faults.

Another major constraint of the previous Approximate Agreement algorithms is that a process can not simply ignore a "missing" value or a self-evident error unless it is assured that all other processes do likewise. Thus, an error can not be classified as benign unless an agreement algorithm is first executed to ensure that all processes agree that it is benign. As a result, previous classes of convergent voting algorithms *can not exploit undiagnosed benign errors*. Furthermore, these restrictions and the facts that agreement algorithms are time consuming and require reliable network communication, may make the benign fault mode an impractical concept. However, the OMSR algorithms by having the ability of ignoring erroneous data items *are able to exploit undiagnosed* benign errors.

The OMSR model can be adapted in any situation in which erroneous values can be detected, due to either self-incriminating errors, error-detecting techniques, or message authentication schemes. For instance, a message may be corrupted by a faulty process or a faulty link. If message authentication is employed, a corrupted message can be detected with a very high probability, and thus be ignored by the receiving non-faulty process. This scenario has the same effect as if no message were received.

Analysis of using same size multisets has the advantage that, for a pair of processes i and j, the corresponding index positions of the selected elements in $\mathbf{M}_i$ and $\mathbf{M}_j$, and the number of elements selected for σ_i and σ_j are the same, i.e. $S_i = S_j$ and $k_i(g) = k_j(g)$. Therefore, *any* selection function will perform *uniformly* on $\mathbf{M}_i$ and $\mathbf{M}_j$. Since this is no longer true in the OMSR model, it was necessary to distinguish among different families of selection functions. Two sub-families were examined: fixed-σ and dynamic-σ. Fixed-σ selection functions utilize a fixed σ regardless of multiset sizes. Dynamic-σ selection functions allow σ to vary as $\mathbf{V}$

changes in size, so that $\sigma_i \neq \sigma_j$. These two sub-families were adopted because together they encompass all commonly used voting algorithms. If $\sigma_{max} = \sigma_{min}$, the fixed-$\sigma$ and dynamic-σ models will yield the same relation $C = (\gamma_{\chi+\omega_{a1}+a})/\sigma$, but will not necessarily produce the same convergence rate because the distribution of the selected elements in each model is different, which may affect γ. On the other hand, if omissions were not allowed to occur, both models converge into the three-mode fault model (MSR) because then all multisets will be of the same size.

Since OMSR has the ability of handling different multiset sizes, it offers potential in examining the convergence process in *irregular* networks, where the nodes may not all be of the same degree. Even if omission faults were not present, its approach to dealing with different multiset sizes can be useful. In addition, in networks where the degree of some nodes are not of sufficient size to meet the lower bound requirements, multiple levels of messages relays can be used to meet the minimum requirements on $V_\circ$ and $|\mathbf{P}_{i \cap j}|$.

Historically, there has been strong dualism in performance and fault-tolerance of Byzantine Agreement and Approximate Agreement. As the algorithms in MSR, the Byzantine Agreement algorithm, known as the *Interactive Consistency* (IC) algorithm, has the same property of substituting a default value for "missing" data items, so that benign faults are hard to justify [23]. For instance, Thambidurai and Park [30] developed a model for IC, in presence of three modes of failure, asymmetric, symmetric, and benign. The result of their analysis is very similar to that of MSR algorithms. Due to their result and other similar studies [12, 23, 27], we conjecture, by using the OMSR methodology, that an OIC (Omission-IC) algorithm can be designed which will have the same advantages as OMSR algorithms, so that omission and undiagnosed benign faults can be exploited.

One area currently under investigation is global convergence for partially connected systems. Although, local convergence is a prerequisite for global convergence, it is not by itself sufficient to guarantee global convergence. The main problem is that, unlike local convergence, it is infeasible, in a large distributed system, to place a limit on the number of faults for each failure mode. Accordingly, examples have shown that if a single non-faulty node diverges, due to exceeding its fault tolerance limit, then the entire system may become non-convergent [20].

References

[1] Anderson K.W., Hall D.W., *Sets, Sequences, and Mappings: the Basic Concepts of Analysis*, Wiley, New York, 1963.

[2] Azadmanesh M.H., Kieckhafer R.M., "The General Convergence Problem: A Unification of Synchronous and Asynchronous Systems", *Dependable Computing and Fault-Tolerant Systems*, Vol. 9, Springer-Verlag, Wien, New York, 1995.

[3] Azadmanesh M.H., Kieckhafer R.M., "A New Hybrid Fault Model for Omissive Faults", Report No. UNO-CS-TR-95-1, Department of Computer Science, University of Nebraska at Omaha, 1995.

[4] Azadmanesh M.H., Kieckhafer R.M., "Hybrid Fault Mode Analyses in Partially Connected Networks", Report No. UNO-CS-TR-95-2, Department of Computer Science, University of Nebraska at Omaha, 1995.

[5] Bartlett J., "A Non-Stop Operating System", *Proceedings of the Hawaii International Conference on System Sciences*, 1978, 103-119.

[6] Babaoglu O., Drummond R., Stephenson P., "The Impact of Communication Network Properties on Reliable Broadcast Protocols", *Proc. 16th Int'l. Symp. on Fault-Tolerant Computing*, Jul 1986, 212-217.

[7] Barborak M., Malek M., "The Consensus Problem in Fault-Tolerant Computing", *ACM Computing Surveys*, Vol. 25, No. 2, June 1993.

[8] Birman K., "Replication and Fault-Tolerance in the ISIS System", *Tenth ACM SIGOPS Symposium on Operating Systems Principles*, New York, Dec 1985, 79-86.

[9] Cristian R., Aghili H., Strong R., Dolev D., "Atomic Broadcast: From Simple Message Diffusion To Byzantine Agreement", *Proc. 15th Int'l. Symp. on Fault-Tolerant Computing*, Jun 1985, 200-206.

[10] Cristian F., Aghili H., Strong R., "Clock Synchronization in the Presence of Omission and Performance Faults, and Processor Joins", *16th Int. Conf. on Fault-Tolerant Computing*, Vienna, Austria, 1986.

[11] Cristian F., Dolev D., Strong R., Aghili H., "Atomic Broadcast in a Real-Time Environment", *IBM Research Report, Almaden Research Center*, 1989.

[12] Dolev D. "The Byzantine generals strike again," *Journal of Algorithms*, Vol. 3, 1982, 14-30.

[13] Dolev D., et al, "Reaching Approximate Agreement in the Presence of Faults," *Proc. Third Symp. on Reliability in Distributed Software and Database Systems*, Oct 1983.

[14] Dolev D., Strong R., "Authenticated Algorithms for Byzantine Agreement", *SIAM Journal of Computing*, Vol. 12, No. 4, 1983.

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

[16] Homes D.C.E., "Global System Data Bus Using the Digital Autonomous Terminal Access Communication Protocol", *Proc. IEEE/AIAA 7th Digital Avionics Systems Conference (DASC)*, Oct 1986, 227–233.

[17] Johnson B.W., *Design and Analysis of Fault-Tolerant Digital Systems*, Addison Wesley, 1988.

[18] Kessels J.L.W., "Two Designs of a Fault-Tolerant Clocking System", *IEEE Trans. Comput.*, Vol. C-33, No. 10, Oct 1984, 912-919.

[19] Kieckhafer R.M., et al, "The MAFT Architecture for Distributed Fault-Tolerance", *IEEE Trans. Comput.*, Vol. C-37, No. 4, Apr 1988, 398-405.

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

[21] Kieckhafer R.M., Azadmanesh M.H., "Low Cost Approximate Agreement in Partially Connected Networks", *Journal of Computing and Information*, Vol. 3, No. 2, 1993, 53-85.

[22] Krishna C.M., Shin K.G., "On Scheduling Tasks With a Quick Recovery From Failure", *Proc. Fifteenth Fault-Tolerant Computing Symposium*, Jun 1985, 234-239.

[23] Lamport L., et al, "The Byzantine Generals Problem," *ACM TOPLAS*, Vol. 4, No. 3, Jul 1982, 382-401.

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

[25] Liu C.L., *Elements of Discrete Mathematics, 2nd Ed.*, New York, McGraw-Hill, 1985.

[26] Lundelius J., Lynch N., "A New Fault-Tolerant Algorithm for Clock Synchronization", *Third Symp. on Principles of Distributed Computing*, Aug 1984, 75-87.

[27] Meyer F.J., Pradhan D.K., "Consensus with Dual Failure Modes", *Proc. Seventeenth Fault-tolerant Computing Symposium*, Jul 1987, 48-54.

[28] Rivest R., Shamir A., Adelman L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *CACM*, Vol. 21, No. 2, 1978, 120-126.

[29] Schneider F.B., *Understanding Protocols for Byzantine Clock Synchronization*, Report No. 87–859, Dept of Computer Science, Cornell University, Aug 1987.

[30] Thambidurai P.M., Park Y.K., "Interactive Consistency with Multiple Failure Modes", *Proc. Seventh Reliable Dist Systems Symp.*, Oct 1988.

[31] Thambidurai P.M., et al, "Clock Synchronization in MAFT", *Proc. Nineteenth Fault-Tolerant Computing Symposium*, Jun 1989, 142-151.

[32] Vasanthavada N., Marinos P.N., "Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Failures", *IEEE Trans. Comput.*, Vol. C-37, No. 4, Apr 1988, 440-448.

[33] Vasanthavada N., Thambidurai P., "Design of Fault-Tolerant Clocks with Realistic Assumptions", *Proc. Eighteenth Fault-Tolerant Computing Symposium*, Jun 1989, 128-133.

[34] Wakerly J., *Error Detecting Codes, Self-Checking Circuits and Applications*, New York, North Holland, 1978.

[35] Wing N.T., *Fault-Tolerant Computing*, Advances in Computers, Vol. 26, 1987, 201-279.

Connectivity	$\tau_\circ$	α	β
Complete	$a+s$	a	$V_\circ + b$
Union	$a+s+f$	$\chi + a$	$V_\circ - \chi$
Intersection	$a+s+\chi$	$\chi + a$	$V_\circ - \chi$

Illustration 1

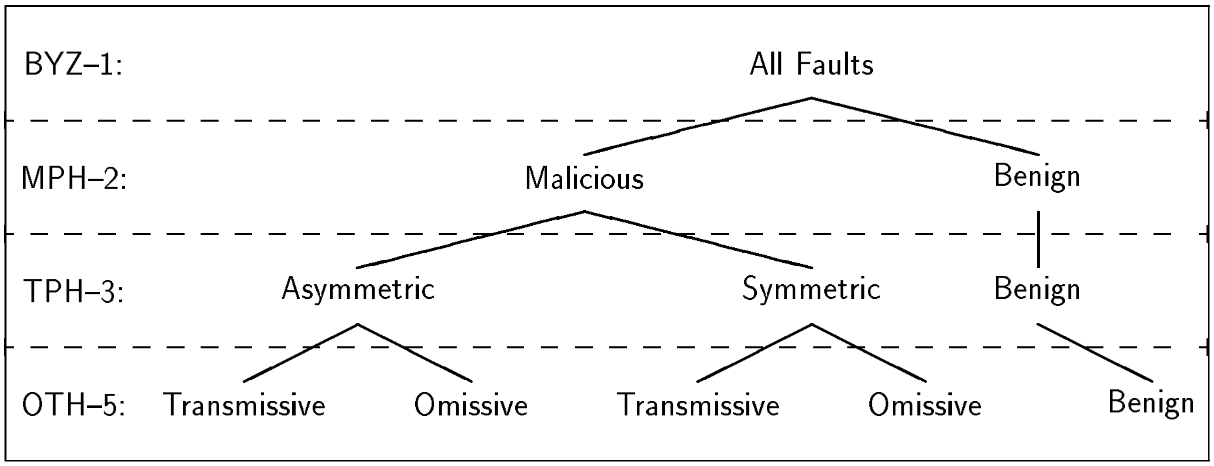
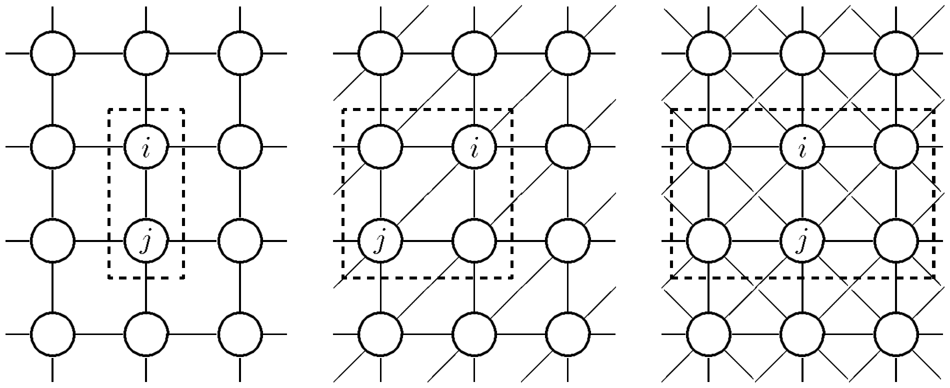


Illustration 2

Connectivity	$\tau_\circ$	α	β
Complete	$a' + s'$	$a' + \omega_{a1}$	$V_\circ + b$
Union	$a' + s' + f$	$\chi + a' + \omega_{a1}$	$V_\circ - \chi$
Intersection	$a' + s' + \chi$	$\chi + a' + \omega_{a1}$	$V_\circ - \chi$

Illustration 3



Quadratic Mesh Hexagonal Mesh Octagonal Mesh

Illustration 4

Table 1: Summary of MSR Convergence Parameters

Figure 1: Relationships Between Fault Models

Table 2: Summary of OMSR Convergence Parameters

Figure 2: Common Mesh Networks

Captions for Illustrations 1 - 4

Authors' biography

Mohammad H. Azadmanesh received the B.S. degree in Cost Accounting from the Iranian Institute of Advanced Accounting, Tehran, Iran, in 1976, the M.S. and Ph.D. degrees in computer science from Iowa State University and University of Nebraska-Lincoln in 1982 and 1993, respectively. He was with Creighton University in the Department of Computer Science/Mathematics, was the Director of Informational Services at Standard Manufacturing Company, and has been consulting for several companies, in Nebraska. He is currently holding a dual appointmentship in the Department of Computer Science and the Center for Management of Information Technology (CMIT) at the University of Nebraska at Omaha, Ne. His research interests include Fault-Tolerant Distributed Systems, Distributed Agreement, Real Time Systems, Network Communications, and Computer Architecture. Dr. Azadmanesh is a member of the Association for Computing Machinery and the IEEE Computer Society.

Axel W. Krings received the Dipl.Ing. in Electrical Engineering from the FH-Aachen, Germany, in 1982 and his M.S. and Ph.D. degrees in Computer Science from the University of Nebraska - Lincoln, in 1991 and 1993, respectively. He is an assistant professor of Computer Science and Computer Engineering at the University of Idaho and a member of the Microelectronics Research Center (MRC), a NASA Space Engineering Research Center. Previous appointments include the Technical University of Clausthal, Germany. His research interests include Fault-Tolerant Systems, Scheduling Theory, Parallel and Distributed Systems, Computer Architecture, and Real-Time Systems. Dr. Krings is a member of the IEEE Computer Society.

Authors' address

Name:	Mohammad H. Azadmanesh
Address:	University of Nebraska at Omaha
	Center for Management of Information Technology (CMIT)
	308J CBA Building
	Omaha, Ne 68182-0459
	USA
Email:	azad@unomaha.edu
Phone:	402-554-3976
Fax:	402-554-3747

Name:	Axel W. Krings
Address:	University of Idaho
	Department of Computer Science
	Moscow, ID 83844-1010
	USA
Email:	krings@cs.uidaho.edu
Phone:	208-885-4078
Fax:	208-885-9052