Approximate Agreements in Multi-Agent Distributed Systems

Kamal Habibi[1], Behnam Zebardast [2], Nazanin Kassaei[3] and Diako Saadi[4]

[1, 2,4]Department of Computer, Boukan Branch, Islamic Azad University, Boukan, Iran
[3]Department of Information Technology, International Campus of Sharif University of Technology Kish Island, Iran

Abstract: *Approximate Agreement is one of the most important issues in Multi-Agent Systems (MAS), in which non-faulty processes execute a voting algorithm to reach on values that are very close to each other. This means that if part of system fails, system can continue its mission. To reach this goal we encounter some challenges in the system such as message dropouts, faults, orchestrated attacks and so on. Thus, this paper has a complete view over all failures that can be organized from different sources and have failure strong impact in the system. Then we will compare three families of approximate agreements called Oblivious, Egocentric and Egophobic under both single-mode and hybrid-fault models based on their convergence-rate.*

Keywords: Approximate Agreement, Hybrid Faults, Byzantine Agreement, Fault-Tolerant.

1. Introduction

The Problem of Distributed Agreement (DA) in Multi-Agent Systems (MASs) has become a research area and has attracted wide range of research fields such as computer science. MASs provide some benefits as following [1, 2]: 1) MASs in networks have computational capabilities and distributed resources. Hence, these kinds of systems are more resistant than Single-Agent System. In other word, there is no Single of Point Failure in MAS. 2) MASs increase the system performance in terms of reliability, extensibility, robustness, maintainability, responsiveness, flexibility and reusability. 3) There is no trust on a single-agent. In a distributed system non-faulty processes have to reach an agreement on data in presence of faulty or Byzantine processes. An error is a part of a system's state that may lead to a failure. Byzantine refers to the Byzantine Generals' Problem, an agreement problem in which generals of the Byzantine Empire's army must decide unanimously whether to attack some enemy army. First time this problem was studied by Lamport et al. in 1982 [3, 4]. They showed in a system that has $\boldsymbol{t}$ faulty processes, agreement can achieve when there are at least $\boldsymbol{2t+1}$ non-faulty processes. Dolev et al. showed that AA problem has to satisfy the following conditions [5]:

- Agreement: All non-faulty processes eventually halt with output values that are within $\boldsymbol{\varepsilon}$ of each other.
- Validity: The value output by each non-faulty process must be in the range of initial values of the non-faulty processes.

In the distributed systems' literature, distributed agreement has been called also as Consensus, Data Fusion, Data Aggregation (DA) and Approximate Agreement (AA) [1, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15]. The paper is organized as follows: Section 2 presents background materials on the evolution of all

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

faults among hybrid fault models. Section 3 presents existing algorithms for reach agreement. Section 4 compares the voting algorithms based on convergence-rate.

2. Background of Faults Models

Faults can be organized from different sources and have failure strong impact in the system. In a system there are several different type faults that can be classified into different levels. Thus, this paper is focused on a mixture of simultaneously present failure impacts, referred to as hybrid fault modeling. Azadmanesh has listed following hybrid fault models.

2.1 BYZ-1: The Single-Mode Byzantine Mode

It's a Single-Mode Fault Model failure, thus, theoretically, it is not classified as a hybrid fault model, but in another models we need its definition. In this type of failure all processes able to have Byzantine behavior. In a synchronous FCN if we do not have condition $\boldsymbol{N \geq 3f + 1}$ consensus will not be held. [16, 17, 18].

2.2 MPH-2: Two-Mode Meyer and Pradhan Hybrid Model

This type of hybrid fault has been presented by Pradhan and Meyer that divide into two distinct modes [19]: 1. Benign: Faults those are self-incriminating or self-evident to all non-faulty processes. 2. Malicious: Faults that do not qualify as benign. The total number of faults in the system, at any given time, is the sum of the number of benign faults,$\boldsymbol{b}$ and the number of malicious faults,$\boldsymbol{m}$ i.e. $\boldsymbol{t = m + b}$.

2.3 TPH-3: Three-Mode Thambidurai and Park Hybrid Model

Thambidurai and Park have presented further partitioned malicious faults into two sub-modes [20]: (1) Symmetric Faults: whose behavior is perceived identically by all non-faulty processes. (2) Asymmetric Faults: whose behavior may be perceived differently by different non-faulty processes. (3) Benign: Faults those are self-incriminating or self-evident to all non-faulty processes. The total number of faults in the system, at any given time, is the sum of the number of benign faults, $\boldsymbol{b}$, the number of malicious faults, $\boldsymbol{m}$ and $\boldsymbol{a}$, asymmetric faults i.e. $\boldsymbol{t = a + m + b}$.

2.4 OTH-4: Four-Mode Omissive/Transmissive Hybrid Model

Azadmanesh and Kieckhafer [21] have proposed a fault model that expands the asymmetric and asymmetric fault modes into transmissive and omissive models. (1) Omissive Symmetric (OS): An OS fault fails to deliver any value to any receiving node. OS faults have many of the same properties as benign faults. (2) Transmissive Symmetric (TS): a TS fault can deliver a single erroneous value to all receiving node. However, by the definition of symmetry, it cannot deliver different values to different processes. (3) Strictly Omissive Asymmetric (SOA): A SOA fault can send a single correct value to some processes and no value to all other processes. However, it cannot transmit an erroneous value to any receiver. (4) Transmissive Asymmetric (TA): A TA fault can exhibit any form of arbitrary asymmetric behavior. Specifically, it can deliver different erroneous values to different receivers. In the OTH-4 model if $\boldsymbol{a'}$ is the number of TA faults $\boldsymbol{s'}$ is the number of TS faults, $\boldsymbol{w_a}$ is the number of SOA faults, and $\boldsymbol{w_s}$ is the number of OS faults in the system, then the total number of faults in the system is given by $\boldsymbol{t = (a' + w_a) + (s' + w_s)}$. The OTH-4 model is also known as AOH-4.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

2.5 PFH-3: Three-Mode Plunkett and Fekete Hybrid Model

Plunkett and Fekete [22] have presented a three-mode fault model consisting of symmetric, asymmetric, and omission faults. The number of omissions in the system is given by the expression $\boldsymbol{O = amax(w_s + w_a)}$, where $\boldsymbol{amax}$ is a function that specifies a priori assumed maximum value. The total number of faults in the system is bounded by the expression $\boldsymbol{t < a' + s' + O}$.

2.6 OTH-5: Five-mode Omissive/Transmissive Hybrid Model

The OTH-5 model is an extension of the OTH-4 model by including the benign faults the total number of faults in the system can be expressed as $\boldsymbol{t = (a' + w_a) + (s' + w_s) + b}$ [22].

Choosing an appropriate fault model for a system depends on system requirements and design perspective. A system may not have a particular fault model or a failure mode during the course of system execution, but we have to consider the worst case behaviors. For example, using the TPH-3 model there may not be any asymmetric faults present. Hence the model would degrade to a two-failure model [22]. Figure 1 shows different types of faults.

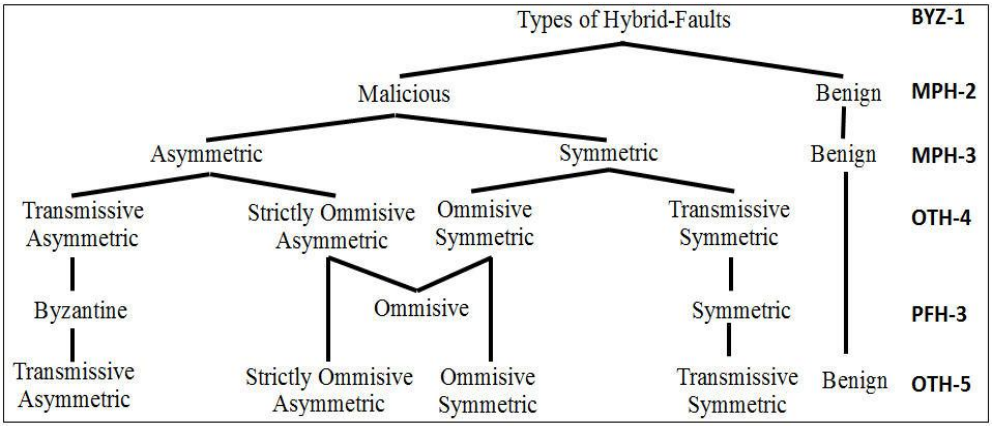

Figure 1. Different Types of Faults [22].

3. Related Works

Turek and Shasha have shown that reaching agreement is only possible for the situations shown in Table 1. In all other cases, it can be shown that no solution exists. Note that most distributed systems in practice assume that processes behave asynchronously, message transmission is point-to-pint, and communication delays are unbounded. As a consequence, we need to make use of ordered (reliable) message delivery, such as provided as by TCP [23].

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

Table 1. Situations under which distributed agreement can be reached [23].

Processors	Message Order				Communication
	Unordered		Ordered		
Asynchronous	No	No	Yes	No	Unbounded
	No	No	Yes	No	Bounded
Synchronous	Yes	Yes	Yes	Yes	
	No	No	Yes	Yes	Unbounded
	Point-to-Point		Broadcast	Point-to-Point	
	Transmission				

3.1 Basic Definitions

A convergent voting algorithm for execution requires a multiset of real value. 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 finite multiset $\mathbf{V}$ of real values may be represented as mapping $\mathbf{V}: \boldsymbol{R} \rightarrow \boldsymbol{N}$. For each real value $\boldsymbol{r}, \boldsymbol{V}(\boldsymbol{r})$ is defined as the multiplicity for $\boldsymbol{r}$ in $\mathbf{V}$. The size of $\mathbf{V}$ is $\boldsymbol{V} = |\mathbf{V}| = \sum_{r \in R} \boldsymbol{V}(\boldsymbol{r})$ [15]. To formally define the criteria for network convergence, the following definitions are needed:

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

$\boldsymbol{max}(\mathbf{V}) = \boldsymbol{max}(r \in \boldsymbol{R}: \boldsymbol{V}(\boldsymbol{r}) > \mathbf{0}) = \boldsymbol{v_1}$; The maximum value of the elements in $\mathbf{V}$.

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

$\boldsymbol{\delta}(\mathbf{V}) = \boldsymbol{max}(\mathbf{V}) - \boldsymbol{min}(\mathbf{V}) = \boldsymbol{v_V} - \boldsymbol{v_1}$; $\boldsymbol{\delta}(\mathbf{V})$ is called the $\boldsymbol{diameter}$ of $\mathbf{V}$.

$\boldsymbol{mean}(\mathbf{V})$ = The arithmetic mean of the real values of all elements of $\mathbf{V}$; $\boldsymbol{mean}(\mathbf{V}) = \frac{\mathbf{1}}{\boldsymbol{V}}\left(\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{V}} \boldsymbol{v_i}\right)$.

$\boldsymbol{S}(\mathbf{V})$ = The multiset remaining after one occurrence of the smallest element of $\mathbf{V}$ has been removed; $\boldsymbol{S}(\mathbf{V}) = \mathbf{V} - \{\boldsymbol{min}(\mathbf{V})\} = \boldsymbol{v2}, \boldsymbol{v3}, \ldots, \boldsymbol{vV}$.

$\boldsymbol{L}(\mathbf{V})$ = The multiset remaining after one occurrence of the largest element of $\mathbf{V}$ has been removed; $\boldsymbol{S}(\mathbf{V}) = \mathbf{V} - \{\boldsymbol{max}(\mathbf{V})\} = \boldsymbol{v1}, \boldsymbol{v2}, \ldots, \boldsymbol{v(V-1)}$.

$\boldsymbol{Red}(\mathbf{V})$ = The multiset remaining after the smallest and the largest element in $\mathbf{V}$ has been removed; $\boldsymbol{Red}(\mathbf{V}) = \boldsymbol{S}(\boldsymbol{L}(\boldsymbol{V})) = \boldsymbol{L}(\boldsymbol{S}(\boldsymbol{V})) = \boldsymbol{v2}, \boldsymbol{v3}, \ldots, \boldsymbol{v(V-1)}$.

$\boldsymbol{V} = |\mathbf{V}_k|$. If less than $\boldsymbol{V}$ values are received, then an arbitrary default value is chosen for each missing value. Thus, $\boldsymbol{V}$ is identical for all non-faulty processes.

3.2 Distributed Algorithms

A variety of convergent voting algorithms have been published which employ the iterative

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

approach of successively receiving data elements and voting on the received elements. In a synchronous network each round follows a number of phases: 1) Broadcast- Each node broadcasts its current value to every node including itself. 2) Collect- Each node collects the values broadcast by other processes including itself. 3) Sample: Each node filters the values to produce the voting multiset $\mathbf{V}$. 4) Execute: The approximation-function $\boldsymbol{F}(\mathbf{V})$ is applied to generate a single voted value. Generally voting algorithms divide into several broad categories, depending on the sampling method used. Oblivious, Egocentric and Egophobic. The family of oblivious algorithms does not place any preference on any input value. In other words, such algorithms are oblivious to the state or source of input values. In egocentric algorithms, a node during the sampling step, favors values that are closer to its own or using its own value when possible to remove the impact of the faulty processes. On the other hand, a node using an egophobic algorithm places favoritism on values that are further away from its own. Such algorithms have been used for hardware clock synchronization [24, 25, 26, 27 and 28].

3.2.1 MSR Algorithms

One family of algorithms that belongs to the oblivious category is called Mean-Subsequence-Reduced or MSR. MSR algorithms can use any value as a default. Therefore, the default value is inherently erroneous. Convergence properties have been determined for an entire family of voting algorithm with the approximation function $\boldsymbol{F}(\mathbf{V}_i) = \boldsymbol{mean}[\boldsymbol{sel}_\sigma(\boldsymbol{Red}^\tau(\mathbf{V}_i)]$,
under a hybrid fault environment consisting of asymmetric, symmetric and benign fault modes. $\boldsymbol{Red}^\tau$ removes the τ largest and τ smallest values from $\mathbf{V}_i$, providing a reduced multiset called medial multiset $\mathbf{M}_i$. The purpose of the $\boldsymbol{Red}^\tau$ function is to mask the impact of erroneous values, so that the voted value will be within the range of correct values. The lower bound on τ is the number of erroneous values and depends on the fault model. $\boldsymbol{sel}_\sigma$ Selects a submultiset, called a subsequence, of σ elements from the medial multiset. The voted value is the arithmetic mean of the subsequence. Therefore, $\boldsymbol{F}(\mathbf{V}_i)$ is the Mean of the Subsequence of the reduced multiset (MSR) [13]. Members of the MSR family differ from each other only in their definition of the selection function $\boldsymbol{sel}_\sigma$. Examples of MSR algorithms include the Fault-Tolerant Midpoint, the Fault-Tolerant Mean, Dolev's optimal algorithm [14], the Mixed-Mode Optimal algorithm, the Binary Mean, the Binary Suboptimal Algorithm [21] and several algorithms for synchronizing phase-locked loops [22].

3.2.2 MSE Algorithms

Azadmanesh and Krings have presented a new family of voting algorithms called Mean-Subsequence-Egocentric (MSE) [29]. MSE algorithms belong to the Egocentric category. Some members of this family have been used in clock synchronization. Thus, a certain amount of discrepancy between processors has to tolerant. This amount of discrepancy depends on application and upper bound on φ is known. During of sampling phase each processor removes the globally diagnosed errors and uses its values as a default for each any value that differs from its own or missing data item by an amount greater than the threshold φ.
During the execution phase, each node executes the approximation function $\boldsymbol{F}(\mathbf{V}) = \boldsymbol{mean}[\boldsymbol{sel}_\sigma(\mathbf{V})]$.
$\boldsymbol{F}(\mathbf{V})$ Is the Mean of Subsequence of an Egocentric (MSE) voting multiset? The varying parameters of $\boldsymbol{F}(\mathbf{V})$ are the number of selected elements σ, and the distribution of these elements in $\mathbf{V}$. Thus, MSE algorithms differ from each other only in their definition of $\boldsymbol{sel}_\sigma$.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

They considered the convergence rate under different fault-models and voting algorithms. Depending on the select elements, three forms of MSE algorithms, differentiated by numeral subscripts, are shown the following:

- $\mathbf{MSE_1}$: For two arbitrary non-faulty processes $\boldsymbol{i}$ and$\boldsymbol{j}$, $|\mathbf{V}_i| = |\mathbf{V}_j|$. Let $\boldsymbol{N}$ be the total number of processes in the systems, and let $\boldsymbol{n} = (\boldsymbol{N} - \boldsymbol{b})$ be the number of data elements after the globally diagnosed benign errors are removed, so that $|\mathbf{V}_i| = \boldsymbol{n}$ for any arbitrary non-faulty node$\boldsymbol{i}$. Thus, the $\boldsymbol{N} - \boldsymbol{b}$ indicates that $\mathbf{MSE_1}$ algorithms select elements which expand over the entire voting multiset.

- $\mathbf{MSE_2}$: In these algorithms the label $\boldsymbol{N} - \mathbf{2}(\boldsymbol{a} + \boldsymbol{b}) - \boldsymbol{b}$ shows that $\mathbf{MSE_2}$ algorithms do not select elements from the extreme $(\boldsymbol{a} + \boldsymbol{s})$ elements of the voting multiset.

- $\mathbf{MSE_3}$: Similarly in these algorithms the label $\boldsymbol{N} - (\mathbf{2}\boldsymbol{a} + \boldsymbol{b}) - \boldsymbol{b}$ indicates the rightmost $(\boldsymbol{a} + \boldsymbol{b})$ and the leftmost $\boldsymbol{a}$ elements are nor considered in the selection process.

3.2.3 MSNR Algorithms

As earlier mentioned MSR algorithms can use any value as a default. Therefore, the default value is inherently erroneous. On the other, the hand, the Mean-Subsequence-Not-Reduced (MSNR) which contain Egocentric and Egophobic categories, effort to replace a missing value or a detected error with a value within the range of correct values. Azad Azadmanesh has shown that these algorithms use the following approximation function: $\boldsymbol{F}(\mathbf{V}_i) = \boldsymbol{mean}[\boldsymbol{sel}_{\sigma}(\mathbf{V}_i)]$.

For each node $\boldsymbol{i}$ to recognizes that a value in $\mathbf{N}_i$ is correct or not, the value has to be apart from node's value by more than a predefined tolerance range $\boldsymbol{\varphi}$. Otherwise the value is accepted as a correct value, even if it is generated by a faulty process. Thus, MSNR algorithms are distinguished from MSR algorithms in tow aspects [24, 26]:

- MSNR algorithms don't use the Reduction function.

- The range of the initial correct values must be known by all processes. Thus, MSNR algorithms don't need to use the Reduction function, whose sole purpose is to generate a submultiset whose range is always within the range of correct values.

3.2.4 MSEP Algorithms

During the sampling, messages may be lost, delayed or corrupted. Thus, a node has to select a new value as a default value to replace with the erroneous messages. The default value by Egophobic algorithms is a value other than the processes' own value. Thus, a node puts more trust on the other value of processes. Among infinite number of acceptable default values that a node $\boldsymbol{i}$ can choose, i.e. from the range $(\boldsymbol{\alpha} - \boldsymbol{\varphi})$ to $(\boldsymbol{\alpha} + \boldsymbol{\varphi})$ the minimum and the maximum acceptable default values, i.e. $(\boldsymbol{\alpha} - \boldsymbol{\varphi})$ and $(\boldsymbol{\alpha} + \boldsymbol{\varphi})$, are most distinguishable [28]. Figure 2 shows family of all voting algorithms.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

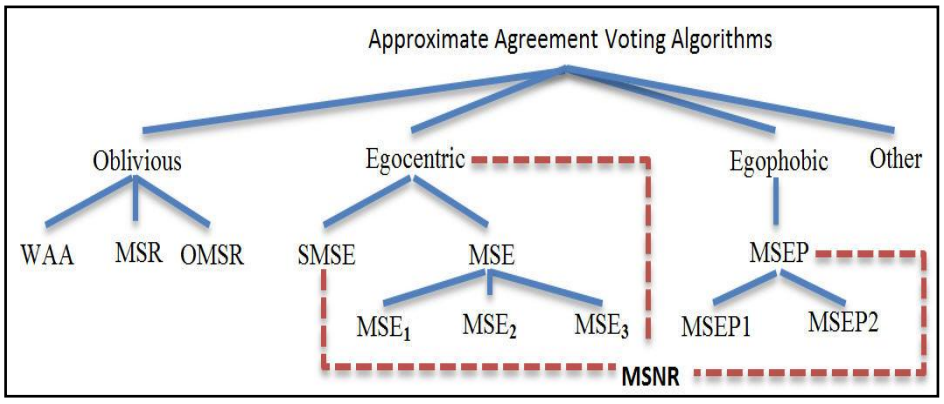

Figure 2. Family of Voting Algorithms.

4. Compression of Voting Models

In this section we compare voting algorithms under both single-mode and hybrid-fault models. Table 2 shows the performance of MSR, MSE, MSNR and MSEP algorithms [28, 24]. We can deduce that given the convergence rate $\mathbf{MSE_3}$ is better than $\mathbf{MSE_2}$ and$\mathbf{MSE_1}$, because $\mathbf{MSE_3}$ has $\boldsymbol{s}$ more elements than $\mathbf{MSE_2}$ for selection and $\mathbf{MSE_2}$ has $\mathbf{2}(\boldsymbol{a}+\boldsymbol{s})$ more elements than $\mathbf{MSE_1}$ for selection, thus, convergence rate will increase. The performance of MSEP algorithms is not better than that of MSR, because the effects of symmetric faults do not distinguish themselves from the effects of asymmetric faults. In comparison to MSE algorithms, it has been shown that MSE algorithms perform their best when no elements from the right $(\boldsymbol{a}+\boldsymbol{s})$ and left $\boldsymbol{a}$ elements are selected. These algorithms perform better than $\mathbf{MSEP_1}$ and $\mathbf{MSEP_2}$ algorithms. Also Table 3 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. MSE and MSR have the same fault tolerance, and their minimum requirement on the number of processes is better than the Byzantine fault model [28].

Table 2. Convergence Rate of MSE Algorithms, where $t=a+b+s$, NC=Not Convergence [25, 26].

	Fault-Model							
Selection Function	**BYZ**	**MSR**	$\mathbf{MSE_1}$	$\mathbf{MSE_2}$	$\mathbf{MSE_3}$	$\mathbf{MSEP_1}$	$\mathbf{MSEP_2}$	**Description**
Mid-point	$\frac{1}{2}$	$\frac{1}{2}$	NC	$\frac{1}{2}$	$\frac{1}{2}$	NC	$\frac{1}{2}$	Selects only the two external values from the medial multiset.
FT-Mean	$\frac{a}{N-2t}$	$\frac{a}{N-2t+}$	$\frac{3a-2s}{N-b}$	$\frac{t-b}{N-2t+}$	$\frac{t-b}{N-2t+s+}$	$\frac{3a-2s}{N-b}$	$\frac{t-b}{N-2t+}$	Selects all the elements from the medial multiset.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

S-Mode Opt	1	$\left\lfloor\frac{N-2t-1}{t}\right\rfloor+1$	1	$\left\lfloor\frac{N-2t+b-1}{t}\right\rfloor+$	3	$\left\lfloor\frac{N-b-1}{t}\right\rfloor+1$	1	$\left\lfloor\frac{N-2t+b-1}{t}\right\rfloor+$	Selects the smallest element from the medial multiset and every t^{th} element thereafter
M-M Opt	1	$\left\lfloor\frac{N-2t-1}{a}\right\rfloor+1$	1	$\left\lfloor\frac{N-2t+b-1}{a}\right\rfloor+$	3	$\left\lfloor\frac{N-b-1}{t-b}\right\rfloor+1$	1	$\left\lfloor\frac{N-2t+b-1}{t-b}\right\rfloor+$	Selects σ contiguous elements of the medical multiset, such that $\sigma=\left\lfloor\frac{N-2t+b-1}{a}\right\rfloor+1$

Table. 3. Comparison of Fault Tolerance among Byzantine, MSR, MSE, and MSEP fault models [27].

Fault Model	Byzantine	MSR	MSE	MSEP
Fault Tolerance	$N \geq 3t+1$	$N \geq 3a+3s+b+1$	$N \geq 3a+2s+b+1$	$N \geq 3a+3s+b+1$

References

1. Azadmanesh, M.H., Kieckhafer, R. M., "Exploiting Omissive Faults in Synchronous Approximate Agreement", *IEEE transactions on Computers*, vol.49, no.10, pp.1031-1042, 2000.
2. Andrew S. Tanenbaum and Maarten van Steen. 2nd ed. (2007). *Distributed systems: principles and paradigms* (pp.1-3, 321-326). Pearson Prentice Hall.
3. Lamport, L., R. Shostak, and M. Pease, 1982, "The Byzantine General Problem", *ACM Transactions on Programming Languages and Systems*, 4, no. 3, July: 382-401.
4. Pease, M., Shostak, T., Lamport, L., "Reaching Agreement in the Presence of Faults", Journal of the ACM, vol. 27, no. 2, pp. 228-234, 1980.
5. Dolev, D., Lynch, N.A., Pinter, S.S., Stark, E.W., Weihl, W.E., "Reaching Approximate Agreement in the Presence of Faults", *Journal of the Association for Computing Machinery*, vol. 83, no. 3, pp. 490-516, 1986.
6. Ren, W., Beard, R.W., Atkins, E.M., Consensus Seeking in Multiprocesse Systems under Dynamically Changing Interaction Topology, *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655-661, 2005.
7. Ren, W., Beard, R.W., Atkins, E.M., A Survey of Consensus Problems in Multi-processe Coordination, Proceedings of the American Control Conference, vol. 3, pp. 1859-1864, 2005.
8. Xiao, F., and Wang, L., "Asynchronous Consensus in Continuous-Time Multi-Processe Systems With Switching Topology and Time-Varying Delays, Proceedings of the IEEE Transactions on Automatic Control, 53 (10): 1804-1816, 2008.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

9. Nakamura, E.F., Loureiro, A.A.F., Frery, A.C., "Information Fusion for Wireless Sensor Networks: Methods, Models and classifications", *ACM Computing Surveys*, vol. 39, no.3, Article 9, pp. 1-55, 2007.
10. Azadmanesh, A., Krings, A.W., and Ghahramani, B., Global Convergence in Partially Fully Connected Networks (PFCN) with Limited Relays, International Journal of Information Technology and Decision Making, vol. 2, no. 2, pp.265-285, 2003.
11. Azadmanesh, M.H., Kieckhafer, R.M., "Asynchronous Approximate Agreement in Partially Connected Networks", International Journal of Parallel and Distributed Systems and Networks, vol. 5, no. 1, pp. 26-34, 2002.
12. Pankaj Jalote. (2007). *Fault Tolerance in Distributed Systems*. USA: Michigan. Prentice Hall, 1994.
13. CRISTIAN, F.: "Understanding Fault-Tolerant Distributed Systems." Commun. ACM, (34)2:56-78, Feb. 1991. Cited on page 324.
14. Hadzilacos, V. and Toueg, S.: "Fault-Tolerant Broadcasts and Related Problems." In Mullender, S. (ed.), Distributed Systems, pp. 97-145. Wokingham: Addison-Wesley, 2nd ed., 1993. Cited on pages 324, 352.
15. Fischer, M., LyncH, N., and Patterson, M. Impossibility of Distributed Consensus with one Faulty Processor:' J.ACM, (32) 2: 374-382, Apr. 1985.Cited on page 334.
16. Kieckhafer, R.M., Azadmanesh, M.H., "Reaching Approximate Agreement with Mixed-Mode faults", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 1, pp.53-63, 1994.
17. Michael Wooldridge. (2002). *An Introduction to MultiProcesse Systems*. (pp.15-17). UK: JOHN WILEY & SONS, LTD, 2002.
18. Meyer, F. J., Pradhan, D.K., "Consensus with Dual Failure Modes", Proceedings of the seventeenth Fault-tolerant computing symposium, pp. 48-54, 1987.
19. Thambidurai, P., Park, Y-K., Interactive Consistency with Multiple Failure Modes, Proceedings of the 7th symposium on Reliable Distributed Systems, pp. 93-100, 1988.
20. Azadmanesh, M.H., and Kieckhafer, R. M., "New Hybrid Fault Models for Asynchronous Approximate Agreement", IEEE Transactions on Computers, vol.45, no.4, pp.439-449, 1996.
21. Plunkett, R., and Fekete, A., Approximate Agreement with Mixed Mode Faults, In the proceedings of the 12th International Symposium of Distributed Computing, pp. 333-346, 1988.
22. Turek, J. and Shasha, S.: The Many Faces of Consensus in Distributed Systems. IEEE Computer, (25)6:8-17, June 1992. Cited on pages 332, 335.
23. Kieckhafer, R.M., Azadmanesh, M.H., "Low Cost Approximate Agreement in Partially Connected Networks", Journal of Computing and Information, vol. 3, no. 1, pp. 53-85, 1993.
24. Lamport, L., and P.M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults", JACM, vol.32, No.1, pp.52-78, Jan 1985.
25. Azadmanesh, M.H., Krings, A.W., Egocentric Voting Algorithms, IEEE Transactions on Reliability , vol. 46, no. 4, pp. 494-502, 1997.
26. Azadmanesh, A, L.Zhou, D. Peak, Egophobic Voting Algorithms, Journal of Computer s & Applications, vol. 25, no. 4, pp. 236-246, 2003.
27. P. Ramanathan, K.G. Shin, R.W. Butler, "Fault-Tolerant Clock Synchronization in Distributed Systems", IEEE Computer, 23, Oct 1990, 33-42.
28. Azadmanesh, A.H.; Sharif, H., "A new look at egocentric algorithms," Parallel and

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)

Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on, p.333, 7-9 July 2004.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-1/MAGNT.21)