

A Confidence-based Software Voter for Safety-Critical Systems

Mohammadreza Rezaee, Yasser Sedaghat, Masoud Khosravi-Farmad
 Dependable Distributed Embedded Systems (DDEmS) Laboratory
 Computer Engineering Department
 Ferdowsi University of Mashhad

mohammadreza.rezaee@stu.um.ac.ir, y_sedaghat@um.ac.ir, masoud.khosravifarmad@stu.um.ac.ir

Abstract— To tolerate software faults, N-Version Programming (NVP) and N-Modular Redundant (NMR) techniques are widely employed. In these techniques, N modules operate on the same data and send their outputs to a software voter. Since the voter is a single point of failure in the techniques, availability and safety are essential requirements. In this paper a confidence-based software voting technique is proposed. The proposed technique considers weight oscillation and confidence of each module to improve availability and safety of the voter. Evaluation results showed that availability and safety of confidence-based software voter in contrast with standard majority voter and adaptive majority voter has improved about 4.4%, 4.9%, and 5.8% for three error injection scenarios.

Keywords—Fault-masking, N-Version Programming (NVP), N-Modular Redundant (NMR), Software Voter.

I. INTRODUCTION

A safety-critical system is a system whose failure may result in financial disaster, death or serious injury to people, or ecological harm [1]. High-speed rail and avionics are good examples for such safety-critical systems which are at least partially controlled by software. Dependability is a major requirement in safety-critical systems which itself requires high reliability. Reliability is the probability (as a function of the time t) that the system has been up continuously in the time interval $[0, t]$ [2]. It is said that a system is failed if its behavior deviates from its intended service [3]. To avoid failures, errors must not occur, and faults should be masked to prevent occurrence of errors. This is the reason fault masking techniques are widely used. Using redundancy is one of the fault masking techniques and is used to achieve safety, availability and reliability. Redundancy can include information redundancy, temporal redundancy or resource redundancy [4]. N-Modular Redundant (NMR) and N-Version Programming (NVP) are examples of using redundancy.

The simplest form of NMR is Triple Modular Redundant (TMR) which is shown in Figure 1. In TMR, three similar modules execute identical operations on the same data and send their outputs to a voter. Voter produces a single output as a final result [5]. A system with fault masking technique may have redundant modules. Voters are used to mask faulty outputs of these modules in every voting cycle [6].

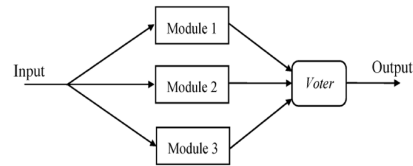


Figure 1 A TMR Voter [5]

The Voter tend to be a single point of failure for most software fault tolerance techniques, so it should be designed and developed to be highly reliable, effective, and efficient [7]. The number of redundant modules in most of safety-critical systems like air traffic control and aircraft control rarely exceeds 5 [8].

Many voting techniques have been proposed in the literature, see [5], [6] and [9], from which the majority voter and its modified versions have been widely used. A majority voter with n inputs produces result if and only if there is an agreement between at least $\lceil (n+1)/2 \rceil$ of its inputs. Otherwise the voter throws an exception and leads the system towards a fail-safe state. Therefore, the majority voter in a TMR system masks a fault in any of three modules in each voting cycle.

It should be mentioned that there are three kinds of faults which may influence the behavior of modules, i.e. permanent faults, transient faults and intermittent faults. Faulty Modules can produce wrong results and send them to the voter [3]. Most of the voting techniques concentrate on masking permanent and transient faults while the proposed technique masks intermittent faults (transient faults which occur frequently) in addition to permanent and transient faults.

Since a voter is a single point of failure in NMR systems, availability and safety are essential requirements. Availability is defined as readiness for correct service and safety means the absence of catastrophic consequences on the user(s) and the environment [10]. In contrast with the existent software voters, this paper presents a software voting technique which has improved availability and safety.

The paper has been organized as follows. In section II, several voting technique are introduced. Section III proposes a new technique that improves majority voter. In section IV the experimental methodology and test results of the proposed method are evaluated. Finally some conclusions are given in section V.

II. RELATED WORK

Voting algorithms have been widely studied in several areas of fault masking systems. Among them, “Unanimity voter” produces results if only all of its inputs are in agreement. The application of this voter is in situations where reaching agreement on all voter inputs is extremely necessary [5]. In order to reduce the severity of this technique, other voting techniques have been introduced. The “majority voter” is one of these techniques that outputs a result where at least $\lceil (N + 1)/2 \rceil$ of its inputs are in agreement, otherwise “No-result” will be produced [7]. In inexact voting, if the difference between voter inputs is smaller than a predefined threshold, these results will be assumed to be in an agreement, while in exact voting, agreement means that the inputs are the same [5]. Frequently, the application of the inexact voting is in software applications which the voter inputs are not exactly the same, even in a fault free environment [5]. An example is software programs that handle floating point calculations. Because there is no mathematical way to define threshold value in inexact voters, this value must be selected carefully through some empirical ways. If the threshold is too small, it leads to many false alarms, while selecting too large values for the threshold avoids false alarms, but the coverage of the fault detection will be reduced due to missing some real faults. “Plural voter” is much like majority voter yet it implements *m-out-of-n* voting, where *m* is less than a strict majority, like as *2-out-of-5* or *3-out-of-7* [5]. “Consensus voter” introduced in [15], is a plurality voting technique which is used in multi-version software with small output space.

Gersting et al. in [9], proposed a majority voter which considers history record of each module as cumulative number of times it is participated in majority agreement in all previous voting cycles. If there is no agreement between results of the modules, this voter chooses the output of the module that has the highest history record. This technique forces the voter to produce a result even in the case of disagreement. In [6] Shabgahi et al. proposed “Adaptive majority voter” which showed using the history record of modules leads to improve the availability and safety of software TMR systems. In each voting cycle (from the moment that voter receives all inputs till the time the result is produced), when there is an agreement between modules results, this voter selects the result of a module which has the highest history record. This module is interpreted as the most reliable one. In cases of disagreement, this technique acts the same as the traditional majority voter.

“Smoothing voter” is a kind of majority voter which is build by adding an acceptance test to the majority voter [8]. Smoothing voter assumes that an error occurs when there is an excessive discontinuity between consecutive module results. In any real-time embedded control applications, which has a feedback control and periodic computation, this assumption is valid. In each voting cycle, in cases of disagreement, this voter selects the closest result to the previous voter output as a probable correct output. This result is taken as the voter output if it has a distance smaller than a pre-defined value named the

“smoothing threshold”; otherwise this voter will not produce any results.

In contrast with the majority voter, which selects a participated input in agreement as the voter output, the “Average voter” calculates a new result which is not among its inputs. This result is the mean of the voter inputs. Moreover, “Weighted average voter” calculates the weighted mean of its inputs. Weight values can be determined statically or dynamically, which have various methods. Several examples about these techniques have been presented in [11] and [12]. The weighted average is calculated using $x_o = \sum w_i \cdot x_i / \sum w_i$ where x_i stands for the voter inputs, w_i represents the weight of module *i*, and x_o denotes the voter output. The precision of the produced results by a weighted average voter depends on the method which determines the weights of modules. One example of a dynamic determination of weights is based on the distance of module results with the calculated weighted average output in each voting cycle. If a module result is far from the calculated weighted average output, it will be assigned a smaller weight compared with a module that its result is closer to the voter output.

“Median voter” selects the median of its inputs as the voted result by a simple algorithm [7]. Majority, Weighted average and Median voting techniques have been generalized to N-Modular Redundant systems in [13].

In several papers, such as [13], [4], [14], [15] and [16], the mentioned voting techniques have been compared. Lorzak et al. in [13] has compared generalized majority, plurality, median and weighted average voters under different failure scenarios and cases that each voter could produce incorrect results are assessed. In cases of disagreement, weighted average and median voters can produce catastrophic errors (incorrect output); in contrast, majority and plurality voters produce exceptions in these cases which may lead the system towards a fail-safe state. In [4], the results of an experimental evaluation of several voters in a variety of simulated error scenarios have been reported. To compare the probability of selecting a correct result by majority, plurality, median and average voters, [14] has derived an expression for each technique. The reliability of consensus, majority and *2-out-of-n* voters are compared in terms of output space cardinality and different values of module reliability in [15]. Parhami in [16] discussed the complexity analyses of different weighted voting algorithms in the worst case.

“Fuzzy voters” are another type of voters that operate based on the distance between voters inputs with each other or the distance between voters inputs and voters output. The fuzzy rules are defined using this distances. [17], [18] and [19] used fuzzy rules to assign weights in weighted algorithms or the majority degree in plurality voters to enhance the accuracy of these voters. These algorithms produce acceptable results. As mentioned in section I, voters are used in safety-critical systems which are mostly real-time systems too. Time overhead of running fuzzy rules may lead the system toward a situation that passes the real-time deadlines, so simpler algorithms are preferred.

The voters are generally divided into two main groups:

- a. The first group includes median, average and weighted average voters, which always produce an output even if their inputs are completely dissimilar.
- b. The voters of the second group, such as smoothing, and adaptive majority voters, generate results if only some of their inputs are similar; otherwise, “No-result” (exception) is generated.

It is obvious that the voters in the first group produce more correct outputs but also more catastrophic outputs than voters in the second group. The voters in the second group generate less incorrect results because of their good error detection capability. Voters in this group produce an output after processing their inputs (e.g., input validation checking, recording module faults, checking value validity, using diagnostic information, using variant fault records). The proposed voting technique in this paper is the combination of the both groups; such that it produces more correct outputs like the first group while it utilizes the employed error detection mechanism in the second group.

III. PROPOSED TECHNIQUE

As mentioned before, in the weighted average voting technique, weight should be considered to reduce the effects of faulty modules on the voter result. If fault occur in a module frequently, the module weight changes repeatedly. To tolerate the effects of intermittent faults, a weighted average voter should consider module weight oscillation.

In a TMR system with an inexact majority voter as shown in Figure 1, a weight w_i is assigned to each module m_i which represents the existence of faults recently affected the module results. At the starting point, a same weight is assigned to all of the modules. If a fault affects a module result so that the module does not participate in the agreement, its weight decreases, while the weight of modules that participated in the agreement increases. If the voter produces “No-result” (there is no agreement), the weights will not change. Changes that are made to weights are predefined. It is important to choose an optimal predefined values for increasing and decreasing weights. The modified weights will not exceed the start (maximum) value and will never be negative values. For an NMR system, the mentioned policy is defined formally as follows:

- a. Sort the voter inputs ascending, at each voting cycle. As a result, for any voter input i (x_i) where $i < n$ then $x_i < x_{i+1}$.
- b. Define a voting threshold (α) and examine the following inequality:

$$x_j - x_i < \alpha \text{ where } i = 1, 2, 3, \dots, n - 1$$

$$\text{and } j = i + 1, i + 2, \dots, n$$

For each i and j that make the inequality correct, α_{ij} is set to 1, which represents that there is an agreement between the module outputs i and j in that voting cycle, otherwise α_{ij} is set to 0.

- c. For each module i , build the following set of $(n - 1)$ Boolean values

$$\{\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in}\} - \{\alpha_{ii}\}$$

Because the module results are sorted, at any point where $\alpha_{ij} = 0$, the rest of $\alpha_{ik}, k > j$ will be 0. For each module m_i , if the value of $\sum_{j=1}^n \alpha_{ij}$ is greater than $(n - 1)/2$, the parameter S_i which indicates that the module m_i has been participated in majority consensus, is set to 1 and its weight increments by a predefined value, otherwise S_i is set to 0 and its weight decrements.

- d. The normalized value $P_i = \frac{1}{n} \cdot \sum_{i=1}^n S_i$ (1) is a measure for reliability level of module i . it is concluded from this formula that a module with a higher P_i has a better performance.

Therewith weight, another parameter called oscillation Osc_i is added to each module, which represents the number of oscillation occurred in the modules weight. Whenever a module weight is increasing or is stable for one or several cycles and just after that this module does not participate in the agreement, its weight decreases, therefore there would be an oscillation in its weight diagram. In this case, Osc_i is decremented by 1. Also there would be an oscillation in case a modules weight acts oppositely, so Osc_i is incremented in this situation too. Figure 2 shows an example of weight diagram. As can be seen in the diagram the module works correctly until column 5. After that its weight decreases because its result does not participate in the agreement so the Osc decreases to 2 (Osc starts at 1). After two cycles that the weight increases, the module participates in agreement again which made the weight to decrease and increase the Osc to 3. The rest of the diagram shows that Ocs decreases with any pulse happening in weight diagram.

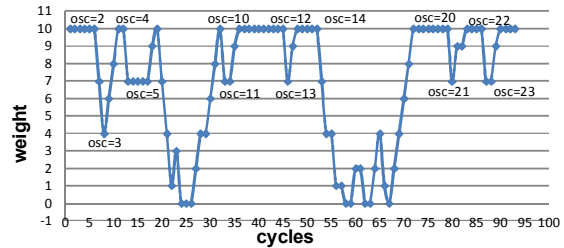


Figure 2 Sample weight diagram

The above strategy is defined formally as follows:

- a. For each modules weight w_i , if it is increasing or is stable on the predefined maximum weight, then f_i is set to 1, otherwise if the weight is decreasing or is stable on the predefined minimum weight, f_i is set to 0. Therefore at any voting cycle, $f_i = 1$ indicates that the module m_i has performed correctly; on the other hand, $f_i = 0$, means that this module has performed incorrectly.
- b. For each module, oscillation Osc_{i+1} is calculated as: ($Osc_1 = 1$)

$$Osc_{i+1} = \begin{cases} Osc_i + 1, & f_i = 1 \text{ and } w_i > w_{i+1} \\ Osc_i + 1, & f_i = 0 \text{ and } w_i < w_{i+1} \\ Osc_i, & \text{Otherwise} \end{cases}$$

The last parameter that is required to calculate the final result of the proposed technique is “confidence”, C_i , which is equal to the weight of the module i , divided by the oscillation of that module. It is obvious that the higher w_i , the higher C_i will be, while the magnitude of Osc_i decreases the value of C_i . So the module with the higher w_i and the lower Osc_i has the highest C_i and is considered as the most reliable module. Confidence in this paper is formally defined as $C_i = w_i/Osc_i$.

As mentioned before, the starting value for Osc_i is 1 and due to the increasing nature of oscillation, Osc_i will never be equal to zero, so the value of the equation will never be undefined. Furthermore, since C_i is calculated only for the modules that are participated in the agreement consensus (which have a non-zero weight), C_i will not be 0 in this equation.

Finally, the outputs of the voter are calculated according to the following steps:

- a. If there is an agreement between the module results, the final result is calculated by the modules which are in the agreement as:

$$Out_i = \frac{\sum_{i=1}^h c_i * r_i}{\sum_{i=1}^h c_i}$$

Where h is the number of modules which participate in the agreement consensus and r_i stands for the result of module i .

- b. If no agreement is detected, the voter throws an exception which will be interpreted as “No-result”. In this case, the values of w_i and Osc_i do not change.

An example of how the proposed technique operates is shown in Table 1. In this example the voter threshold is 0.5 and the correct result in all sets is 1. The start (maximum) weight is set to 10 and weights increase by 2 and decrease by 3. The first majority set found will be the final majority set, if there are more than one (These numbers are just to simplify the test).

IV. EXPERIMENTAL RESULTS

To evaluate the improvement of the proposed voter (confidence-based voter), it is compared with the standard majority voter because it has the same definition of agreement consensus, and the adaptive majority voter [6] because it also applies the history record of modules to

extend the standard majority voter. These voters are placed in a hypothetical TMR system and modules results are sent to these voters. The produced results are then compared together. Similar to previous studies, e.g. [4], [6], [8], to evaluate the voter, modules outputs are produced, using $100 + 100 \sin(t)$ with different t values for each module, so it is possible to inject error (to the notional correct value 100) to modules independently. The injected errors can be controlled by selecting a random t within the interval of $[-\theta + \theta]$. By selecting different values for θ , the error injected to each module can be controlled, so the magnitude of injected errors for different modules varies. Also the injected errors for a module can be various in different cycles to simulate intermittent and transient faults.

In each experiment set including 10000 voting cycles (n), the number of correct results (n_c), the number of incorrect results (n_i) and the number of cycles which produced “No-result” (n_n) is counted. This information is further used to calculate the safety and availability of the voters. As mentioned in Section I, availability is defined as readiness for correct service and safety means the absence of catastrophic consequences on user(s) and the environment. The value $S = 1 - \frac{n_i}{n}$ is taken as the safety index and the value $A = \frac{n_c}{n}$ is defined as the availability index of voters [6]. According to this definition, the ideal voter is a voter with $S = 1$ and $A = 1$.

The accuracy threshold determines if the distance between the notional correct value and the voter output is within acceptable limits. Accuracy threshold is assumed to be 0.5 which is equal to the voter threshold. Note that the voter threshold is maximum acceptable divergence of module outputs from the notional correct value in each voting cycle. Since all three mentioned voters apply the same way in response to disagreement, cases which produce “No-result” are identical. The experimental results also prove this point.

Different experiments with various amount of injected errors to the modules executed, in which f_1 represents the amount of injected error to the first module, f_2 represents the amount of injected error to the second module, etc. The experimental result of voters with various injected error levels: $f_1 \subseteq [-0.5, +0.5]$, $f_2 \subseteq [-1, +1]$, $f_3 \subseteq [-1.5, +1.5]$ is shown in Table 2.

Table 1 Example of Confidence-based Voter

| test set | module results | | | module weights | | | module oscillations | | | module confidence | | | final answer |
|----------|----------------|-----|-----|----------------|----|----|---------------------|------|------|-------------------|----------|----------|--------------|
| | m1 | m2 | m3 | w1 | w2 | w3 | osc1 | osc2 | osc3 | c1 | c2 | c3 | |
| 1 | 1.2 | 1.5 | 1.8 | 10 | 10 | 7 | 1 | 1 | 2 | 10 | 10 | 3.5 | 1.35 |
| 2 | 0.9 | 1.3 | 1.7 | 10 | 10 | 4 | 1 | 1 | 2 | 10 | 10 | 2 | 1.1 |
| 3 | 1.4 | 0.7 | 1.4 | 10 | 7 | 6 | 1 | 2 | 3 | 10 | 3.5 | 2 | 1.4 |
| 4 | 0.8 | 1 | 0.3 | 10 | 9 | 3 | 1 | 3 | 4 | 10 | 3 | 0.75 | 0.846153846 |
| 5 | 0.8 | 1 | 1.6 | 10 | 10 | 0 | 1 | 3 | 4 | 10 | 3.333333 | -- | 0.85 |
| 6 | 1.1 | 1.7 | 1.6 | 10 | 7 | 2 | 1 | 4 | 5 | 10 | 1.75 | 0.4 | 1.119230769 |
| 7 | 0.4 | 1.2 | 1.8 | 10 | 7 | 2 | 1 | 4 | 5 | | | | no-result |
| 8 | 1.3 | 0.4 | 0.2 | 7 | 9 | 4 | 2 | 5 | 5 | 3.5 | 1.8 | 0.8 | 0.338461538 |
| 9 | 1 | 1 | 1.9 | 9 | 10 | 1 | 3 | 5 | 6 | 3 | 2 | 0.166667 | 1 |
| 10 | 1.2 | 1.6 | 1.7 | 10 | 10 | 3 | 3 | 5 | 7 | 3.333333 | 2 | 0.428571 | 1.376033058 |
| 11 | 0.5 | 0.4 | 1.2 | 10 | 10 | 0 | 3 | 5 | 8 | 3.333333 | 2 | --- | 0.4625 |

Table 2 Results of the First Test

| Voting Techniques | Correct Results | Incorrect Results | No Results |
|-------------------|-----------------|-------------------|------------|
| Majority | 2346 | 3128 | 4526 |
| Adaptive Majority | 2510 | 2964 | 4526 |
| Confidence-based | 2753 | 2721 | 4526 |

The results show that the confidence-based voter has more correct and less incorrect results than both standard majority voter and adaptive majority voter.

Another experimental result of voters with different injected error levels: $f_1 \in [-1, +1]$, $f_2 \in [-1.5, +1.5]$, $f_3 \in [-2, +2]$ is as follow:

Table 3 Results of the Second Test

| Voting Techniques | Correct Results | Incorrect Results | No Results |
|-------------------|-----------------|-------------------|------------|
| Majority | 1011 | 3101 | 5888 |
| Adaptive Majority | 1056 | 3056 | 5888 |
| Confidence-based | 1107 | 3005 | 5888 |

As can be seen in the above results, number of disagreed outputs are the same in all three voters, the reason is using the same strategy in response to disagreement by the mentioned voters.

The final results of voters with equal but small injected error levels: $f_1, f_2, f_3 \in [-0.5, +0.5]$ is as follow:

Table 4 Results of the Third Test

| Voting Techniques | Correct Results | Incorrect Results | No Results |
|-------------------|-----------------|-------------------|------------|
| Majority | 4645 | 4602 | 753 |
| Adaptive Majority | 5647 | 3600 | 753 |
| Confidence-based | 6865 | 2382 | 753 |

The results show that all of the voters have a same low probability of reaching disagreement. In this case, 753 number from 10000 number of answers are “No-result” (7.5%), so there is a considerable difference in performance of the voters compared to the previous mentioned experiments.

Another experiment set executed, with a hypothesis that the third module has a higher probability of being exposed by errors than the first two modules. The injected errors amplitude in this experiment set is in the range of $t \in [0.6, 1.4]$ in which always $f_3 = f_1 + 0.2$. The selected error scenarios are: (0.6 0.6 0.8), (0.7 0.7 0.9), (0.8 0.8 1), (0.9 0.9 1.1), (1 1 1.2)... (1.4 1.4 1.6) where the first element of each tuple represents the maximum injected error to the first module, the second element of the tuple represents the maximum injected error to the second module, etc. Figure 3 shows the availability of voters versus injected error amplitude in this experiment set. The safety of voters versus injected error amplitude is shown in Figure 4.

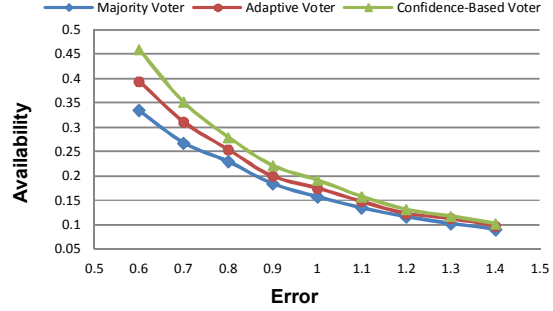


Figure 3 Availability versus Injected Error amplitude

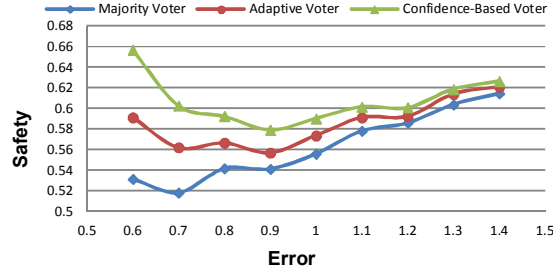


Figure 4 Safety versus Injected Error amplitude

According to section II, transient errors may occur in modules of a TMR system. Another set of experiment with previous assumptions is built to test these kind of errors. At first, equal amount of errors are injected to all modules, and then after running 4000 cycles, the error amplitude of the third module is raised by 0.5 in the next 2000 cycles. Availability of standard majority voter, Adaptive Majority Voter and confidence-based voter, for this scenario, is shown in Figure 5. In Figure 6 safety of standard majority voter, Adaptive Majority Voter and confidence-based voter, for this scenario is shown.

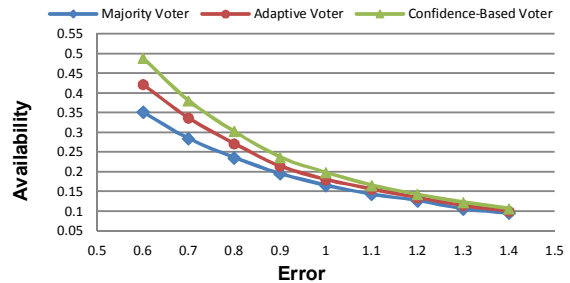


Figure 5 Availability versus Injected Error amplitude for transient errors

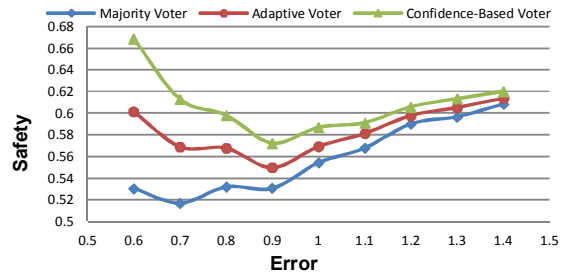


Figure 6 Safety versus Error amplitude for transient errors

The last experiment set is executed to test the effects intermittent errors, which are introduced in section II. The scenario for this test is that, at first, equal amount of error is injected to all of the modules. After that the injected error to the third module is increased and decreased by 0.5 in a 2000 periods. Which means that in the first 2000 cycles, modules have an equal error amplitude, then in second 2000 cycles the error amplitude of the third module is raised by 0.5 and then it is decreased by 0.5, etc.

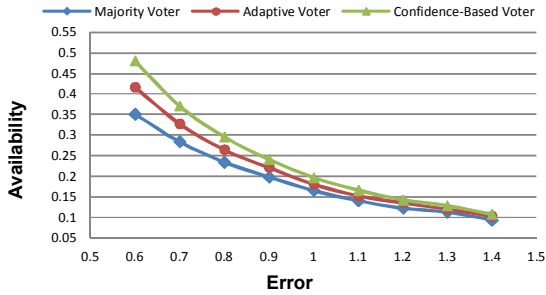


Figure 7 Availability versus Error amplitude for intermittent errors

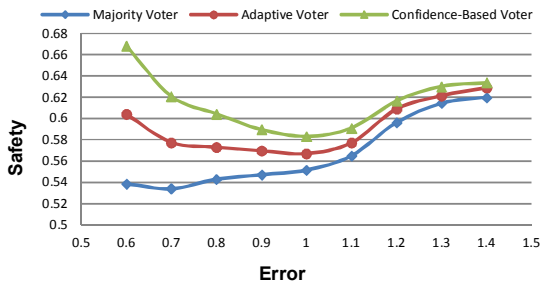


Figure 8 Safety versus Injected Error amplitude for intermittent errors

In the experiments, as the error amplitude increases, the number of “No-result” increases too. As mentioned before $A = \frac{n_c}{n}$ and $S = 1 - \frac{n_i}{n}$ that can be interpreted as $A = 1 - \frac{n_i + n_n}{n}$ and $S = \frac{n_c + n_n}{n}$ which indicate that availability has inverse relationship with the number of “No-result” while safety has direct relationship with number of “No-result”, so as the number of “No-result” increases, the availability decreases and the safety increases.

V. CONCLUSIONS

The paper addressed the benefits of considering oscillation for each modules weight in standard majority voter in a Triple Modular Redundant system. A new technique, “confidence-based majority voter”, is proposed and a method to produce confidence in a majority voter is devised. The technique applies weight and oscillation to mask permanent, transient and intermittent faults that may occur during system’s life cycle. The results show that the proposed technique improves the availability of standard majority voter (4.4%) and safety of standard majority voter (4.4%). In case of occurring transient faults in each module this technique improves the availability and safety of standard majority voter by 4.9%. If a system is exposed to intermittent faults, this technique improves the availability and safety of standard majority voter by 5.8%. In cases of disagreement, this technique behaves the same

as the standard majority voter. The experimental results demonstrate that the proposed voter is superior in safety and availability against standard majority and adaptive majority voters in different error scenarios.

REFERENCES

- [1] J.C. Knight, “Safety critical systems: challenges and directions”, *Proceedings of the 24th International Conference on Software Engineering (ICSE,2002)*, pages 547 – 550, 2002.
- [2] Koren, I. and C.M. Krishna, *Fault-Tolerant Systems*, Elsevier, 2007.
- [3] Kopetz, H., *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Springer, 2nd Edition, 2011.
- [4] J. M. Bass, G. Latif-Shabgahi, and S. Bennet, “Experimental comparison of voting algorithms in case of disagreement”, *Proceedings of the 23rd EUROMICRO Conference, New Frontiers of Information Technology*, pages 516 – 523, 1997.
- [5] G. Latif-Shabgahi, J.M. Bass, and S. Bennett, “A Taxonomy for Software Voting Algorithms Used in Safety-Critical Systems”, *IEEE Transactions on Reliability*, Vol. 53 , Issue 3, pages 319 – 328, 2004.
- [6] G. Latif-Shabgahi, and S. Bennett, “Adaptive Majority Voter: A Novel Voting Algorithm for Real-Time Fault-Tolerant Control Systems”, *Proceedings of the 25th EUROMICRO Conference*, Vol. 2 , pages 113 – 120, 1999.
- [7] Pullum, L.L., *Software Fault Tolerance Techniques and Implementation*, Artech House, 2001.
- [8] G. Latif-Shabgahia, S. Bennett, and J.M. Bass, “Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems”, *Journal of Microprocessors and Microsystems*, Vol. 27, Issue 7, Elsevier, pages 303–313, 2003.
- [9] J.L. Gersting, R.L. Nist, D.B. Roberts, and R.L. Van Valkenburg, “A comparison of voting algorithms for n-version programming”, *Proceedings of the 24th Annual Hawaii International Conference on System Sciences*, Vol. 2, pages 253 – 262, 1991.
- [10] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing”, *IEEE Transactions on Dependable and Secure Computing*, Vol. 1 , Issue 1, pages 11-33, 2004.
- [11] R.B. Broen, “New voters for redundant systems”, *Journal of Dynamic Systems, Measurement and Control*, pages 41–45, 1975.
- [12] Z. Tong and R. Kain, “Vote assignments in weighted voting mechanisms”, *Proceedings of the 7th Symposium on Reliable Distributed Systems*, pages 138–143, 1988.
- [13] P.R. Lorzczak, A.K. Caglayan, and D.E. Eckhardt, “A theoretical investigation of generalised voters”, in *Digest of Papers FTCS’19: IEEE 19th Annual International Symposium On Fault-Tolerant Computing Systems*, pages 444–451, 1989.
- [14] D.M. Blough and F.G. Sullivan, “A comparison of voting strategies for fault-tolerant distributed systems”, *Proceedings of 9th IEEE Symposium on Reliable Distributed Systems*, pages 136–145, 1990.
- [15] D.F. McAllister, C. Sun, and M.A. Vouk, “Reliability of voting in fault tolerant software systems for small output space”, *IEEE Transactions on Reliability*, Vol. 39 , Issue 5, pages 524–533, 1990.
- [16] B. Parhami, “Voting Algorithms”, *IEEE Transaction on Reliability* Vol. 43 , Issue 4, pages 617–629, 1994.
- [17] G. Latif-Shabgahi, A.J. Hirst, “A fuzzy voting scheme for hardware and software fault tolerant systems”, *Fuzzy Sets and Systems*, Volume 150, Issue 3, 16 March 2005, pages 579-598.
- [18] Ondrej Linda, Milos Manic , “Interval Type-2 fuzzy voter design for fault tolerant systems”, *Information Sciences*, Journal of Elsevier Volume 181, Issue 14, 15 July 2011, pages 2933-2950.
- [19] Singamsetty, PhaniKumar and SeethaRamaiah Panchumarthy. "Automatic Fuzzy Parameter Selection in Dynamic Fuzzy Voter for Safety Critical Systems" *IJFSA* 2.2. pages 68-90, 2012.