Available online at www.sciencedirect.com

SCIENCE DIRECT®

Microprocessors and Microsystems 27 (2003) 303–313

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems

G. Latif-Shabgahi[a,*], S. Bennett[b], J.M. Bass[c]

[a]Telematics Department, Technology Faculty, The Open University, Walton Hall, Milton Keynes MK7 6AA, UK
[b]Department of Automatic Control and System Engineering, The University of Sheffield, Mappin Street, Sheffield S1 3JD, UK
[c]Training and Development Consultant, Merit International Inc., London, UK

Received 16 August 2002; revised 16 February 2003; accepted 18 February 2003

Abstract

Voting algorithms are used to arbitrate between variant results in a wide range of highly dependable real-time control applications. These applications include N-Modular Redundant hardware systems and diversely designed software systems based on N-Version Programming. The most sophisticated and complex voting algorithms can even tolerate malicious (or Byzantine) variant errors. Voting algorithms can be implemented in either hardware or software depending on the characteristics of the application and the type of voter selected. The behaviour of voting algorithms in multiple error scenarios is considered in this article. Complete disagreement is defined as those cases where no two variant results are the same. A novel algorithm for real-time control applications, the smoothing voter, is introduced and its behaviour compared with three previously published voters. Software implemented error-injection tests, reported here, show that the smoothing voter achieves a compromise between the result selection capabilities of the median voter and the safety features of the majority voter. The smoothing voter is an appropriate voter for applications in which maximising the number of correct outputs and minimising the number of benign errors of the system is the main concern, and a slight degradation in safety can be tolerated.
© 2003 Published by Elsevier Science B.V.

Keywords: Fault-tolerance; Availability; Safety; Safety-critical system

1. Introduction

Many applications can exhibit hazardous behaviour in the presence of faults. There is considerable economic, social and moral pressure to minimise the probability of dangerous system behaviour. Industrial sectors, which employ such systems, include process control (for example where toxic, flammable or explosive materials are used), transportation (such as railway, avionics and increasingly, automotive systems), nuclear power station and military applications. In these applications the goal is increasing system dependability [11], often in the presence of conflicting trade-offs. Fault masking is one of the primary approaches to improve or maintain the normal behaviour of a system in a faulty environment [9]. N-modular redundancy and N-version programming [7] are the well-known fault masking methods. These approaches use redundant modules (variants) and a voting unit to hide the occurrence of error(s) from the system output. The simplest form of N-modular redundancy approach is Triple Modular Redundancy, TMR, as shown in Fig. 1. Three similar modules perform an identical function on the same input data and deliver the results to the voter. The voter arbitrates between the achieved results and produces a single output. The number of redundant modules in practical cases such as safety-critical systems (e.g. aircraft control, nuclear power plant control) rarely exceeds 5. There are situations, however, where voting with a fairly large number of inputs is required. One example is in image processing filters where during each pass pixel values may be replaced by values determined from voting on a predefined neighbourhood of nearby point [6].

The problem of voting on inexact results arises when correctly functioning modules produce slightly different results due to, for example, rounding or truncation errors. Inexact voters may use thresholding techniques to define

* Corresponding author. Tel.: +44-1908-653359; fax: +44-1908-65-3658.
E-mail address: g.r.latif@open.ac.uk (G. Latif-Shabgahi).

0141-933/03/$ - see front matter © 2003 Published by Elsevier Science B.V.
doi:10.1016/S0141-9331(03)00040-1

Nomenclature

n	Number of tests
n_c	Number of a voter correct outputs during n runs
n_{ic}	Number of a voter incorrect outputs during n runs
n_d	Number of a voter benign errors during n runs
ε	Threshold for inexact majority voter
β	Smoothing voter threshold
A_T	Accuracy threshold of the test harness
e_{max}	Maximum amplitude of injected errors
MAJ	Majority voter
MED	Median voter
WA	Weighted average voter
SM	Smoothing voter

boundaries within which agreement can be said to exist. Results that lie outside the *threshold*, or lie a large distance away from some average value, are not considered to be in agreement. Complete disagreement is defined as those cases where no two-module results are the same. Related work on voting algorithms is presented in Section 2. A novel voting technique, the *smoothing voter*, which is convenient for use in real-time computer-based control systems with a cyclic input stream, is introduced in Section 3. The software implemented error-injection test harness is discussed in Section 4 and experimental results are presented in Section 5. Conclusions of the work are given in Section 6.

2. Related works

2.1. Voting algorithms

Voting algorithms are required to identify those variant results in agreement and select one of these as the voter output. Several well known voting algorithms have been widely used in commercial applications. The n-input majority voter produces a correct result where at least $[(n+1)/2]$ voter inputs match each other. In cases of no majority, the voter generates an exception flag, which can be detected by the system supervisor to move the system toward a safe state. The formalised plurality voter implements m-out-of-n voting, where m is less than a strict majority (e.g. 2-out-of-5 voting). The median voter is a mid-value selection algorithm. This algorithm successively eliminates pairs of outlying values until a single result remains (the algorithm assumes an odd number of redundant inputs). The weighted average voter, on the other hand, calculates the weighted mean of its redundant input values. Weight values can be determined in various methods; see for example, Refs. [5,12,17]. Calculated weights, w_i, are then used to compute the voter output, $z = \Sigma_{w_i \cdot x_i} / \Sigma_{w_i}$ where x_i values are the voter inputs and z is the voter output. One example is the distance metric based weighted average voter in which the weight values are dynamically calculated based on the distances between the module results in each voting cycle. A module result, which is far away from the other module results, is assigned to a smaller weight compared with a result that is close to any of the other module results. Thus, the algorithm does not select a result from the voter inputs, but instead produces a new result. The majority, median and weighted average algorithms have been generalised to **N**-Modular Redundant systems by Lorczak [14].

Step-wise negotiated voting has been developed for a long-life space application [10]. Diagnosis information is obtained from periodic self-test operations. However, the self-test information is used to supplement voting in cases of disagreement voting cycles. In this voter the diagnostic information is given priority, i.e. in cases of disagreement the self-test results are used to select a result. The paper has shown that step-wise negotiated voting offers improvements over independent majority voting or the stand-by redundancy technique [9]. An optimal voter can be proposed, where the probability of error arrivals is known (or can be assumed) [4]. Thus, an ideal voter can be described mathematically. However, in real systems exact knowledge of error arrival probabilities is not known. Thus, the optimal voter output for any given set of variant results is also unknown. The problem of exact (or bit-by-bit) voting on vectors of information has been investigated by Gersting [8]. A key issue is the resolution of conflicting agreements between the different fields within a set of vectors. The solutions proposed include the calculation of a composite vector rather than selection of any of the input vectors and the use of application specific weightings to minimise the importance of information in less important vector fields. Parhami [16] considered the performance, in terms of execution time, of voters, and proposed efficient implementations of a variety of algorithms. Embedded control applications are typically cyclic systems in which there exists some relationship between the result in one cycle and the result in the next. Knowledge of this relationship between successive results is used in predictive voters [1] to produce results in cases of

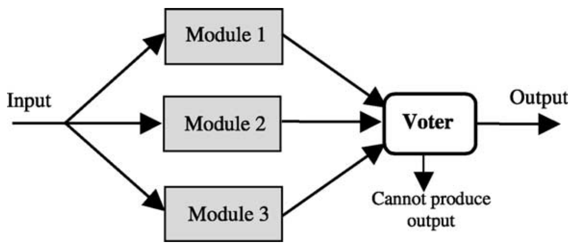

Fig. 1. A triple modular redundant system.

disagreement. A history of previous voter results is used to generate an expected result value where disagreement is detected. The expected result is compared with each of the variant results in order to make a selection. A comprehensive survey and taxonomy for software voting algorithms can be found in Ref. [13].

2.2. Comparison of voters

Several works have contrasted the behaviour of generic and purpose-built voters. The cases in which each algorithm is capable of generating incorrect results have been tabulated in Ref. [14]. The authors show that the majority and plurality algorithms can produce exceptions (i.e. benign as opposed to catastrophic errors) where disagreement is detected. The weighted average and median voters, in contrast, can produce catastrophic errors in such cases of disagreement. In Ref. [4] the probability of choosing a correct value by majority, plurality, median and mean voting algorithms, under a probabilistic model, has been investigated. For each voter an expression for the probability of choosing the correct value has been derived and these expressions are then used to compare the voters. A comparison between the execution time of a multi-stage NMR system with various voting algorithms including simple majority voter, majority Expedient Voter without runahead and majority Expedient Voter with runahead has been performed in Ref. [18]. In Ref. [15] the reliability comparison of majority, 2-*out-of-n*, and consensus voters in terms of various values of variant reliability, output space cardinality has been provided. The complexity analysis of various weighted voting algorithms in the worst case has been discussed in Ref. [16].

There is relatively little published work contrasting voter behaviour in multiple error scenarios [3]. Many authors take the view that multiple simultaneous errors are sufficiently unlikely that they can be ignored. However, such multiple error scenarios can produce catastrophic failures and should therefore be analysed. In safety-critical systems any catastrophic failure is unacceptable and potential causes should be explored. The error detection capabilities of the majority and plurality voters are obtained at a price. It is easy to imagine double simultaneous errors (in a triple modular redundant configuration) which can be masked by selecting the mid-value variant result. The median voter, in particular, is shown to be successful at masking such double errors. The smoothing voter has been designed to achieve a compromise between the error detection capabilities of the majority voter and the results selection capabilities of the median voter.

The empirical study, reported in Sections 4 and 5 later, contrasts the behaviour of algorithms in these multiple error cases. Firstly a new algorithm is introduced in Section 3, the smoothing voter, whose behaviour has been specifically designed considering complete disagreement cases.

3. Smoothing voter

3.1. Definition

This algorithm extends the majority voter by adding a special kind of acceptance test. The acceptance test is based on the assumption that a discontinuity between consecutive variant results is indication of an error. This assumption is valid in many real-time embedded control applications where there is feedback control and periodic computation. In the smoothing voter, when there is no agreement between variant results, the closest result to the previous voter output is selected as the probable output for this cycle. If the measured distance is smaller than a pre-defined value named as ‘smoothing threshold’, then that result is taken as the voter output; otherwise, the voter fails to produce answer. It is assumed that the first cycle of algorithm is successful and its output is available for next cycle. The selection of a value for the smoothing threshold parameter of this algorithm is critical. Although arbitrary values can be used, improved performance will be obtained if information about the probable discontinuity size of consecutive results of the system during its mission time is available. This algorithm can be defined more formally as follows:

S1. Let $A = \{d_1\ d_2\ d_3 \dots d_n\}$ denote the set of N variant results

S2. Sort the set A in ascending order to construct the new set $AS = \{x_1\ x_2 \dots x_n\}$

S3. Construct the following partitions from AS, for all $j = 1 : m$, in which $m = (N+1)/2$:

$$V_j = \{x_j\ x_{j+1} \dots x_{j+m-1}\}$$

S4. If at least one of the partitions $V_j (j = 1, 2, \dots m)$ satisfies the property $d(x_j, x_{j+m-1}) \geq \varepsilon$, then the majority for original set A is satisfied. Here ε is the voter threshold and d is a distance metric between elements.

S5. If none of the partitions V_j satisfies the property $d(x_j, x_{j+m-1}) \geq \varepsilon$, then determine the output x_k in A such that $d(x_k, X) = \min\{d(x_1, X), d(x_2, X) \dots d(x_n, X)\}$, where X is the previous successful voter result.

S6. If x_k satisfies the condition $d(x_k, X) \leq \beta$, where β is the smoothing threshold, then select x_k as the voter output, otherwise, the voter has not found agreement.

3.2. Basic smoothing voter

The algorithm defined in Section 3.1 is called *Basic smoothing voting algorithm* in which the smoothing threshold β is fixed. In Table 1 the results of this voter for a stream of inputs including seven samples have been

Table 1
Basic smoothing alg. ($\varepsilon = 0.1, \beta = 1$)

Sample	var1	Var2	var3	result
1	1	1.1	1.4	1.1
2	2.1	2.3	3	2.1
3	3.1	3.2	7	3.1
4	4.2	4.9	5.5	No result
5	5	5.3	5.8	No result
6	6	6.8	6.5	No result
7	7	7.1	7.5	7

Table 2
Modified smoothing alg. ($\varepsilon = 0.1, \beta = 1$)

Sample	var1	var2	var3	Result
1	1	1.1	1.4	1.1
2	2.1	2.3	3	2.1
3	3.1	3.2	7	3.1
4	4.2	4.9	5.5	No result
5	5	5.3	5.8	5
6	6	6.8	6.5	6
7	7	7.1	7.5	7

illustrated. It is assumed that voter threshold ε is equal to 0.1 and smoothing threshold β is 1. In the first sample, agreement has been found and the result of variant one has been selected as voters' output. There is no majority for the second sample; therefore according to S5 the closest variant output of this cycle to the previous successful output of voter, 2.1, is chosen. Note that the difference between this selected output and previous output of voter is smaller than β. The output of third cycle is 3.1 due to the existing majority between variant results. Again, in cycle 4, there is no agreement, the candidate output is 4.2 (closest one to the previous output of voter) which does not satisfy the necessary condition $d(4.2, 3.1) < \beta$. Thus a *no vote exception* is raised in sample 4. The following two cycles have similar situations while for 7th sample a majority is achieved. There is a problem, which occurs with this fixed smoothing threshold. As stated, at sample 4 there is disagreement between var1, var2 and var3 and now var1 differs by 1.1 from previous value hence no output is produced. Since no output is produced the reference for future computations remains 3.1 produced at sample 3. Hence the voter cannot produce an agreed output until two of the variants satisfy the basic threshold β. This is shown at sample 7.

3.3. Modified smoothing voter

A more rapid recovery from a transient error is achieved, by the introduction of a cumulative smoothing threshold which is dynamically adjusted after each cycle where *no result* is found. Where each successive variant result is more positive than its predecessor then the initial smoothing threshold β is added to the cumulative smoothing threshold. In contrast, where each successive variant result is more negative than its predecessor then the initial smoothing threshold is subtracted from the cumulative smoothing threshold. The behaviour in this case is shown in Table 2.

At sample 4 no *result* has been produced, then the value of β has been changed to a new threshold, 2β, to be used in the following cycle. At sample 5 there is again disagreement between variants output but the candidate output, var1 = 5, has satisfied the condition $d(5, 3.1) < 2\beta$. Hence the result 5 has been selected for this cycle; also the value of 2β has been changed to its initial value. By using this strategy there is no problem for sample 6 to produce an output. If β is selected very large then the smoothing algorithm approaches the performance of the median voter.

4. Experimental method

4.1. Assumptions

The experimental work reported here is based on the following assumptions and terminology:

- the voter is used in a cyclic system where there exists some relationship between correct results from one cycle to the next;
- that faults cause errors whose symptoms appear to the voter as numerical input values perturbed by varying amounts;
- that perturbations (below some predefined accuracy threshold) in voter input values are considered as acceptable inaccuracies;
- that large perturbations in voter input values are considered as errors;
- for the purposes of the results reported here, the issues associated with ensuring synchronisation of the inputs to the voter (or indeed, synchronisation of the inputs to variants) is ignored;
- there exists some 'notional correct result', which can be calculated from the current inputs and system state, based on the history of previous system states;
- the 'notional correct result' is the desired voter result even where voter inputs are erroneous;
- a comparator is used to check for agreement between the notional correct result and the voter output; and
- an accuracy threshold is used, in the comparator, to determine if the distance between the notional variant result and the voter output is within acceptable limits.

These assumptions are representative of a class of voters common in embedded real-time control systems. The assumptions exclude those applications where voter inputs are randomly distributed through the input space in successive cycles.

4.2. Error model

Each voter input is defined as a member of a tuple; $\{c, \varepsilon\pm, A_T \pm\}$; in application specific value units. Where, c is the notional correct value of each voting cycle, $\varepsilon+/\varepsilon-$ is the predefined upper/lower voter threshold (in those voters that depend on thresholds to perform inexact voting), and A_T+/A_T is the upper/lower accuracy threshold.

This model represents a simplified version of the error model presented in Ref. [2]. The voter is tested in a triple modular redundant configuration, as shown in Fig. 2, in these experiments.

One notional correct result is produced by the input generator in each test cycle. This sequence of numbers simulates identical correct results generated by redundant variants. Copies of the notional correct result are presented to each saboteur, in every cycle. The saboteurs can be programmed to introduce selected variant error amplitudes, according to selected random distributions. In a given set of tests one, two or three saboteurs may be activated to simulate variant result errors on the voter inputs.

4.3. Experimental scenario

The voters tested are: formalised traditional voters (majority voter, plurality voter, median voter and weighted average voter) and the novel smoothing voter. The performance of voters is dependent on many parameters such as: input profile, execution-time, injected error amplitude (in comparison with the voter threshold), error distribution, number of perturbed inputs, the value of voter threshold, accuracy-threshold and smoothing-threshold for the smoothing voter. Various scenarios can be constructed by different combination of these parameters. The parameters used for the experiments reported below are as follows:

1. Input to variants: sinusoidal function $u(t) = 100\cdot\sin(t) + 100$ sampled at 0.1 *sec*.
2. Voter-threshold value, $\varepsilon = 0.5$.
3. Accuracy-threshold value, $A_T = 0.5$.
4. Two/three variants were perturbed using uniform error distribution with amplitude varies from 0.5 to 10.
5. Smoothing threshold (for the smoothing voter) is set to a fixed value.
6. Amplitude of errors varies from 0.25 to 5% of the peak input signal value to cover applications with low-faulty modules to those with high-faulty modules.

4.4. Performance criteria

The selected performance criteria are: normalised correct results (the ratio of correct outputs to the whole number of runs; n_c/n), normalised incorrect results (the ratio of incorrect results to the total number of tests,; n_{ic}/n) and normalised disagreed outputs (the ratio of disagreed results to the total number of runs; n_d/n). These measures may be expressed in percentage form.

The measure n_c/n represents the capability of a voter in producing correct results. It can be considered as an appropriate criterion to evaluate voter *availability*; higher values of this measure are desirable. The ratio n_{ic}/n is a measure of catastrophic outputs. From a safety point of view the smallest number of incorrect outputs which is consistent with small ratio of n_{ic}/n is desirable. This ratio can be a good performance criterion to evaluate a voter used in a *safety* critical system in which incorrect results are unacceptable. Finally the ratio of disagreed results of a voter to the total number of runs, n_d/n, represents the capability of a voter in producing benign results. These are errors successfully detected by the voter, which can be used to alert some higher-level error management system. It must be noted that a 3-version programming system using inexact voter may produce *no-result* (a special code) output in the cases that (i) the voter can not reach consensus upon the results of variants (such cases occur in majority and plurality voters) or (ii) the built-in mechanism within a voter cannot produce an acceptable result in disagreement cases (for example, in smoothing voter, the smoothing mechanism can not

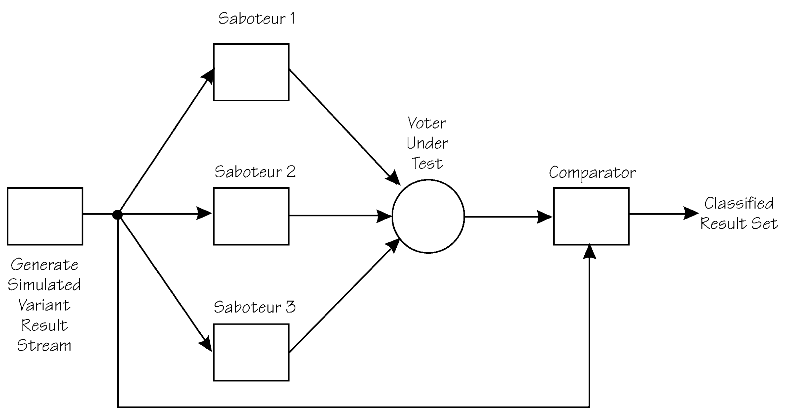

Fig. 2. Experimental test harness.

generate a reasonable result). The *no-result* output can be used to initiate a safe shutdown of the system. It is obvious from safety viewpoint that generating a *no-result* output is much better than producing agreed but incorrect results. Hence, a tuple ($n_c/n, n_{ic}/n, n_{na}/n$) describes the performance of selected voters in this paper.

5. Experimental results

The results of voter comparison based on selected experimental scenarios defined in 4.3 in terms of performance criteria defined in 4.4 are presented. Since the behaviour of majority and plurality voters in a three version programming system is similar, only results for a majority voter are given. Each voter has been tested for 10^4 runs with fixed accuracy threshold equal to 0.5.

5.1. Experiment 1. Results using triple error injection

In this experiment, for each voter all of the inputs are perturbed to examine the voter behaviour in the worst case. Fig. 3 shows a comparative plot of n_c/n for voters versus the error-amplitude for a wide range of errors. The figure indicates a non-linear (a negative exponential, approximately) relation between the increase in the amplitude of errors and decrease of n_c/n. The median and weighted average voters have a similar performance with the median voter giving a slightly higher ratio of n_c/n than the weighted average voter. Similarly, there is little to distinguish the majority and smoothing voters although the smoothing voter gives slightly a better ratio of n_c/n than the majority voter.

Fig. 4 highlights the plot of the ratio of $n_c/n$of voters for small errors. The median voter has 2–15% better performance than the both majority and smoothing voters while the behaviour of the majority and smoothing voters are very similar.

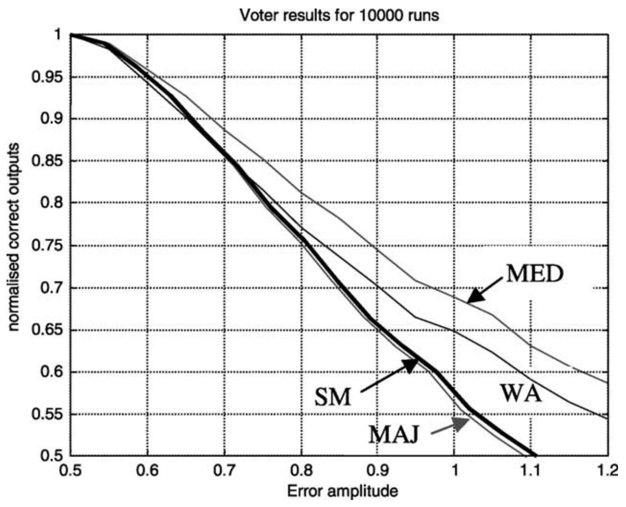

Fig. 4. Normalised correct results for triple error injection with small errors.

Fig. 5 shows the plot of n_{ic}/n of the compared voters versus the error amplitude for 10^4 runs. For all voters, the number of incorrect outputs increases with the error amplitude. However, above certain error amplitude, the majority and smoothing voters begin to detect some voting cycles as complete disagreement cases, and hence disregard the corresponding inputs toward in producing the voter output. Therefore, the number of incorrect outputs decreases, and the number of benign outputs increases. The figure shows that, for all errors, the majority voter produces the smallest number of incorrect outputs whereas the weighted average voter gives the largest number of incorrect results. The median voter has a similar behaviour to the weighted average voter, and the performance of the smoothing voter lies between that of the majority and weighted average voters. The large number of incorrect outputs for median and weighted

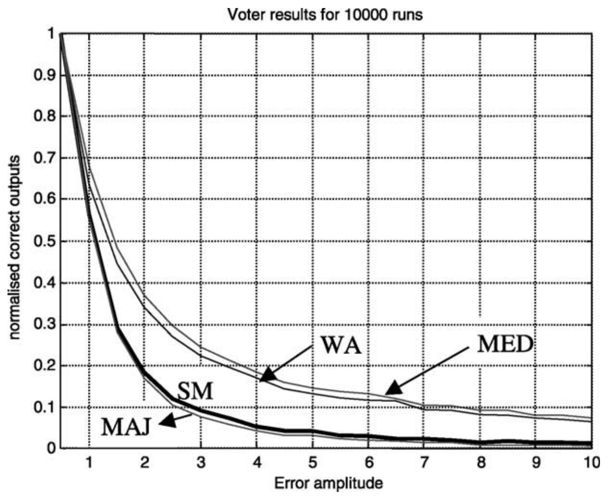

Fig. 3. Normalised correct results for triple error injection.

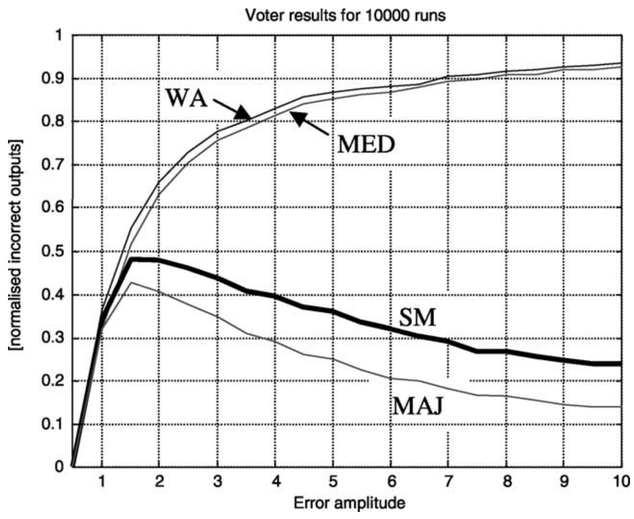

Fig. 5. Normalised catastrophic results for triple error injection.

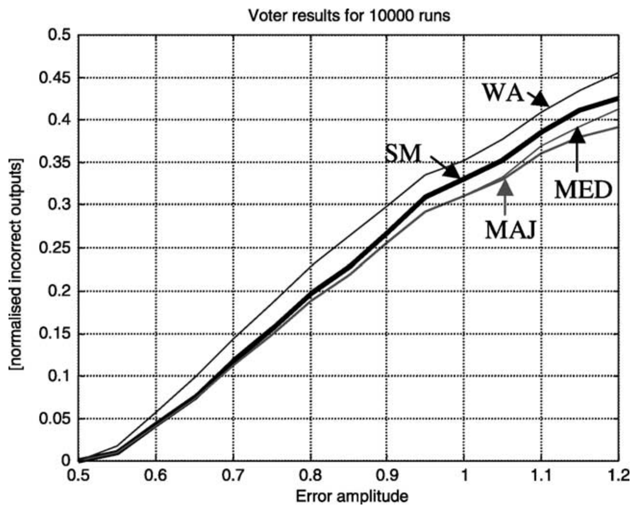

Fig. 6. Normalised catastrophic results for triple error injection with small errors.

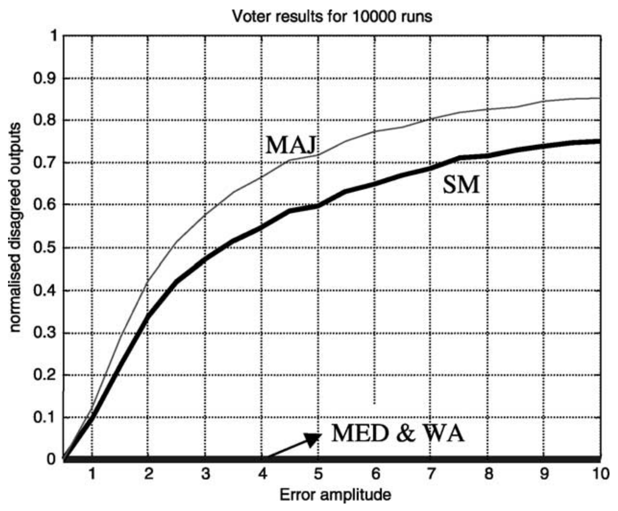

Fig. 7. Normalised begin results for triple error injection.

averaging voters limits their application in situations where large errors are likely to occur and safety is the main concern.

It should also be noted that for inexact voters the possibility of reaching an incorrect agreement on redundant inputs perturbed by small errors is higher than that on inputs perturbed by large errors. This is why the ratio of n_{ic}/n for majority voter at $e_{\max} = 1$ is larger than that at point $e_{\max} = 7$, for example.

Fig. 6 highlights the behaviour of voters in terms of the ratio of n_{ic}/n for small errors (where $e_{\max} <\approx 1.2$). The median voter performs as well as the majority voter with the lowest ratio of n_{ic}/n, the weighted average voter gives the highest ratio, and the performance of the smoothing voter is between the majority/median and weighted average voters. From Figs. 4 and 6 we conclude that in the presence of small-size permanent triple errors, the median voter, in spite of its simplicity, gives a better performance (more correct and less incorrect outputs) than both the majority and weighted smoothing voters. Therefore, the smoothing voter is not the most suitable voter for small permanent multiple errors.

Figs. 3 and 5 well illustrate two main groups of voters:

1. The first group including median and weighted average voters always produces output regardless of the validity of their inputs. This group generates larger number of incorrect results.
2. Group two includes voters that produce an output after some sort of processing voter inputs (value validation checks, consensus checking, etc.). This group generates smaller number of incorrect results compared with the first group.

Fig. 7 shows a plot of normalised disagreed (benign) results of voters; n_{na}/n, versus error amplitude. The median and weighted averaging voters do not produce benign outputs; this makes them unsuitable for the safety critical systems. As expected, the majority voter produces more benign outputs than the smoothing voter; $n_{\mathrm{d}}(\mathrm{MAJ}) > n_{\mathrm{d}}(\mathrm{SM})$. The reason is that the smoothing voter, by means of its built-in mechanism, is capable of handling more multiple error cases than the majority voter.

If at any error point we define $q[= n_{\mathrm{d}}(\mathrm{MAJ}j) - n_{\mathrm{d}}(\mathrm{SM})]$, then for determining the effectiveness of the smoothing voter over the majority voter, we should investigate what portion of q is handled correctly (to result in a correct voter-output), and what portion is handled incorrectly (to result in an incorrect voter-output) by the voter. Figs. 3 and 5 show that a large portion of q is handled unsuccessfully (wrongly); that is, the number of benign cycles (among q) that are handled incorrectly by the smoothing voter (and as a result, an incorrect voter-output is produced) is more than those handled successfully resulting in a correct voter output. In other words, for all error cases $n_{ic}(\mathrm{SM})/q > n_{\mathrm{c}}(\mathrm{SM})/q$. Table 3 shows the numerical values for some of the selected error cases for clarification of this point. In this table, $n_{\mathrm{ic}}(\mathrm{SM})$ stands for the number of incorrect outputs, and $n_{\mathrm{c}}(\mathrm{SM})$ shows

Table 3
Capability of Smoothing voter in handling q benign Results in selected error scenarios based on Figs. 3 and 5

Parameters	$e_{\max} = 1$	$e_{\max} = 2$	$e_{\max} = 3$	$e_{\max} = 4$	$e_{\max} = 5$
q	278	841	1056	1169	1217
$n_{\mathrm{ic}}(\mathrm{SM})$	216(77%.q)	683(81%.q)	903(85%.q)	1057(90%.q)	1104(91%.q)
$n_{\mathrm{c}}(\mathrm{SM})$	62(23%.q)	158(19%.q)	153(15%.q)	112(10%.q)	113(10%.q)

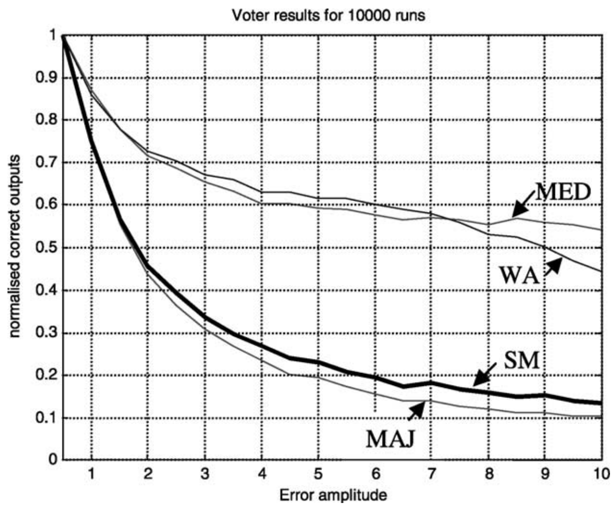

Fig. 8. Normalised correct results for double error injection.

the number of correct outputs of the smoothing voter. Note that, for example, where $e_{max} = 1$, of the 278 benign cases, the smoothing voter produces a correct output in 62 cases, and gives an incorrect output in 216 cases. More of the outputs are incorrect than correct. The same occurs for other error points in the table. This suggests that the use of the smoothing voter, as a replacement for a majority voter, in the presence of permanent triple errors is not beneficial; it converts most of the benign cases of the majority voter to incorrect outputs, and thus threatens the system safety.

A similar set of experiments, carried out with normally distributed errors, showed that in this case all the voters produce more correct outputs and less catastrophic and benign results than in the previous experiments. The reason is that in the normal case, due to the shape of error distribution, the probability of having perturbed variant results with small amplitude of error is more than that of the uniformly distributed case. In fact, the experiment with uniformly distributed error is carried out under worse conditions than a normal scenario.

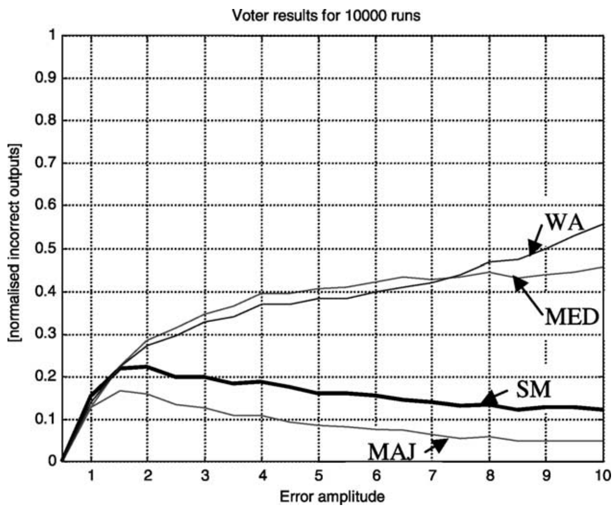

Fig. 9. Normalised incorrect results for double error injection.

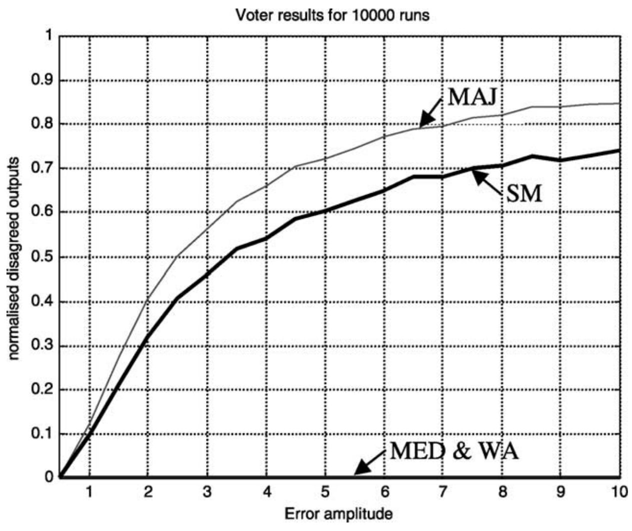

Fig. 10. Normalised benign results for double error injection.

5.2. Experiment 2. Results of double error injection

In this experiment, only two voter inputs are perturbed with permanent errors. Figs. 8–10 show the comparative results versus error-amplitude for a wide range of errors. Compared with the results of the previous experiment in this experiment, all of the voters give higher ratio of n_c/n and lower ratio of n_{ic}/n as expected. The relative behaviour of voters (to each other) is similar to the previous experiment with one exception, the weighted averaging voter gives a higher ratio of n_c/n than the median voter in many error intervals (where $1.5 < e_{max} < 7$). The smoothing voter gives a considerably higher ratio of n_c/n than the majority voter (Fig. 8), shows a conservative performance between the baseline voters in terms of the ratio n_{ic}/n (Fig. 9), and gives a lower ratio of n_d/n than the baseline voters (Fig. 10). Although its performance is better than that of the previous experiment (especially from the viewpoint of n_c/n), it still converts most (about 70%) of the benign cases of the majority voter to incorrect than correct outputs (See Table 4). Again, for small errors, the median voter is the best candidate (plots are not shown here).

5.3. Experiment 3. Results with double transient errors

In this experiment we examine the behaviour of voters in a more realistic error case. Since in reality, the probability of occurrence of transient errors is more than that of the permanent (or intermittent) errors, the performance investigation is carried out with transient errors. The impact of transient errors is simulated by injecting two arbitrary values from the interval $[-e_{max} \dots + e_{max}]$ to randomly selected saboteurs every T_e voting cycle. The values of T_e as well as the e_{max} are adjustable; the results reported here are achieved with a T_e randomly selected from interval [10,15] and with errors with e_{max} from interval $[-10 + 10]$. Fig. 11–13 show the comparative results of voters.

Table 4
Capability of smoothing voter in handling q benign results in selected error scenarios based on Figs. 8 and 9

Parameters	$e_{max} = 1$	$e_{max} = 2$	$e_{max} = 3$	$e_{max} = 4$	$e_{max} = 5$
q	278	850	1019	1155	1147
n_{ic}(SM)	246(88%.q)	642(75%.q)	735(72%.q)	794(68%.q)	771(67%.q)
n_c(SM)	32(12%.q)	208(25%.q)	284(28%.q)	361(32%.q)	376(33%.q)

The most benefits of the smoothing voter are appeared in this case; it gives less benign outputs, considerably more correct outputs, and slightly more incorrect outputs than the majority voter in most error cases. In other words, it successfully converts a large portion of the benign outputs of the majority voter to a correct output, and increases considerably the system availability while slightly decreases its safety. For example, at $e_{max} = 2$, in which $q = 100$, more than 75% of the benign outputs of the majority voter are handled and successful converted to correct outputs by the smoothing voter (in this case, the voter outputs are: Majority $[n_d = 346, n_c = 9526, n_{ic} = 128]$, and Smoothing $[n_d = 246, n_c = 9600, n_{ic} = 154]$).

The results of testing the voters by injecting normally distributed transient errors (double injection) revealed the capability of the smoothing voter for better performance (results not shown). It is concluded that the novel smoothing voter is an invaluable voter for handling transient errors; those errors that occur most frequently if software and hardware systems, their appropriate handling can considerably improves the system performance.

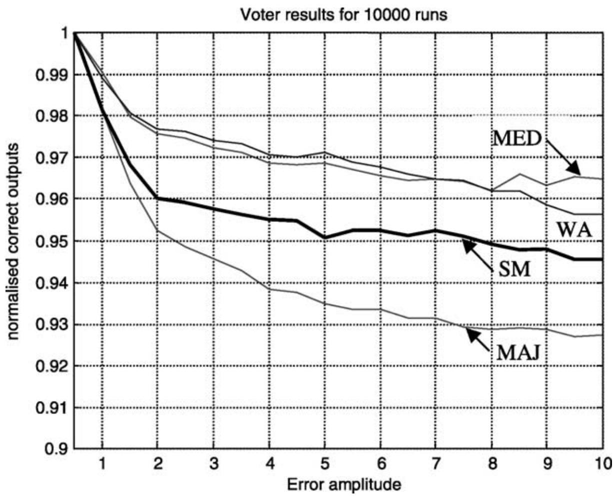

Fig. 11. Normalised correct results for transient errors.

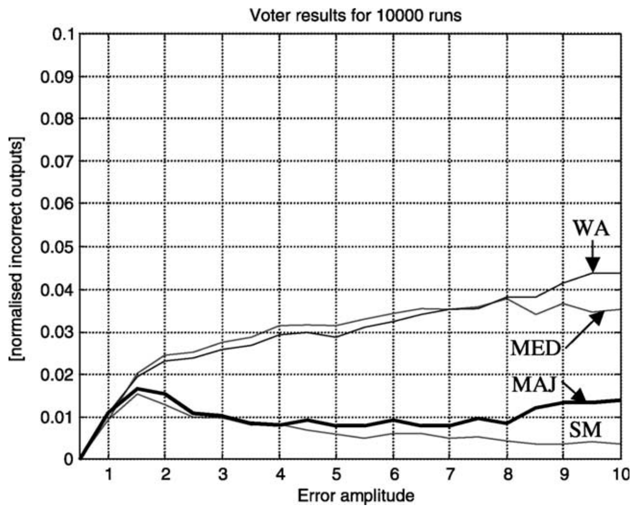

Fig. 12. Normalised incorrect results for transient errors.

6. Conclusions

The empirical behaviour of four three-input voting algorithms in complete disagreement cases (double/triple error scenarios) has been evaluated in this paper. It was shown that the majority voter has conservative error detection capability in such scenarios. The median and weighted averaging voters, in contrast, can mask certain classes of double error, but at the expense of producing catastrophic errors in triple error scenarios. The smoothing voter is introduced with the objective of reaching a compromise between the result selection capabilities of the median voter and the conservative error detection properties of the majority voter. The results, presented in Section 5, show that the novel smoothing voter produces less benign errors and more correct results than the majority voter. In addition, the smoothing voter produces less catastrophic errors than the median voter. Moreover, we

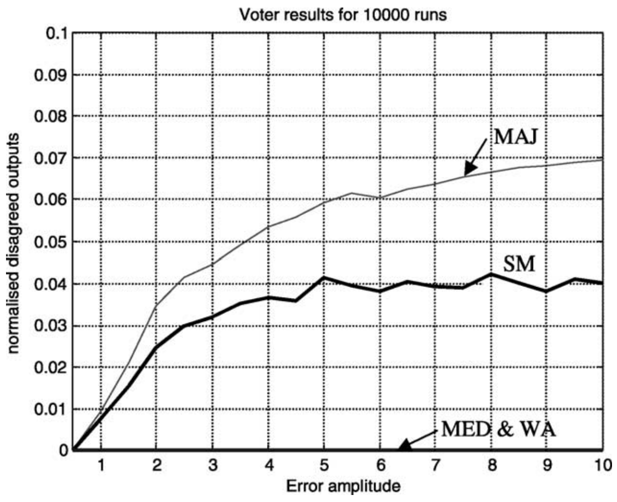

Fig. 13. Normalised benign results for transient errors.

showed that with triple injection of permanent errors the smoothing voter decreases the number of benign outputs of the majority voter at the cost of a large increase of its incorrect outputs, and a small increase of its correct outputs. However, when faced with transient errors, the smoothing voter gives more correct than incorrect outputs (among the benign cycles it handles). Since in reality, the probability of occurrence of transient errors is more than that of the permanent errors, we conclude that the smoothing voter is an appropriate voter for applications in which improving the system availability is the main concern at the cost of decreasing the number of benign errors and by definition a slight decrease in the system safety.

wSelection of an appropriate voter for a specified system depends on the nature and requirements of that application. When, for example, a large number of correct outputs is required (e.g. availability is the main concern), the median voter is the best candidate to employ. When an application requires the least possible incorrect outputs (e.g. safety is the main concern), the majority voter is preferred. A trade off between these two cases can be achieved by means of the smoothing voter introduced in this paper.

Acknowledgements

The authors gratefully acknowledge the useful comments of Dr Tony Hirst from the Telematics Department of the Open University as well as anonymous referees.

References

[1] J.M. Bass, Voting in Real-Time Distributed Computer Control Systems, PhD Thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, 1995.

[2] J.M. Bass, P.J. Fleming, A.M. Tyrell, An error model for computer control systems, UKACC International Conference on Control, Exeter (1996) 353–358.

[3] J.M. Bass, G. Latif-Shabgahi, S. Bennett, Experimental comparison of voting algorithms in cases of disagreement, Proceedings of 23rd Euromicro Conference, Budapest, Hungary (1997) 516–523.

[4] D.M. Blough, F.G. Sullivan, A comparison of voting strategies for fault-tolerant distributed systems, Proceedings of IEEE Ninth Symposium on Reliable Distributed Systems (1990) 136–145.

[5] R.B. Broen, New voters for redundant systems, Journal of Dynamic Systems, Measurement and Control (1975) 41–45.

[6] D.R.K. Brownrigg, The weighted median filter, Communication of the ACM 27 (8) (1984) 807–818.

[7] L. Chen, A. Avizienis, N-Version programming: a fault-tolerance approach to reliability of software operation, Digest of Papers FTCS'8: IEEE Eighth Annual International Symposium on Fault-Tolerant Computing Systems, Toulouse, France (1978) 3–9.

[8] J.L. Gersting, A Comparison of voting algorithms for N-version programming, Proceedings of 24th Annual Hawaii International Conference on System Sciences 2 (1991) 253–262.

[9] B.W. Johnson, Design and Analysis of Fault-Tolerant Digital Systems, Addison-Wesley Publishing Company, USA, 1989.

[10] K. Kanekawa, H. Maejima, H. Kato, H. Ihara, Dependable On-Board Computer Systems with a new method: stepwise negotiated voting, Digest of Papers FTCS'19: IEEE 19th IEEE 19th Annual International Symposium on Fault-Tolerant Computing Systems, Chicago (1989) 13–19.

[11] J.-C. Laprie, Dependable computing and fault-tolerance: concepts and terminology, Digest of Papers FTCS'15: IEEE 15th Annual International Symposium on Fault-Tolerant Computing Systems, Ann Arbor, Michigan (1985) 2–11.

[12] G. Latif-Shabgahi, Performance Analysis of Software Implemented Inexact Voting Algorithms, PhD Thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK, 1999.

[13] G. Latif-Shabgahi, J.M. Bass, S. Bennett, A taxonomy for software voting algorithms used in real-time embedded control systems, to appear in IEEE Transactions on Reliability (2003).

[14] P.R. Lorczak, A.K. Caglayan, D.E. Eckhardt, A theoretical investigation of generalised voters, Proceedings of IEEE 19th International Symposium On Fault-Tolerant Computing Systems (1989) 444–451.

[15] D.F. McAllister, C. Sun, M.A. Vouk, Reliability of voting in fault-tolerant software systems for small output space, IEEE Transactions on Reliability 39 (5) (1990) 524–533.

[16] B. Parhami, Voting Algorithms, IEEE Transaction on Reliability 43 (4) (1994) 617–629.

[17] Z. Tong, R. Kain, Vote assignments in weighted voting mechanisms, Proceedings of the Seventh Symposium on Reliable Distributed Systems (1988) 138–143.

[18] M.A. Vouk, A.M. Paradkar, D.F. McAllister, Modelling execution time of multi-stage N-version fault-tolerant software, Proceedings of the IEEE Computer Software and Application Conference (1990) 505–511.

Dr G. Reza Latif-Shabgahi, Telematics Dept., Technology Faculty, The Open University, Milton Keynes, MK7 6AA, UK. Internet (e-mail): g.latif@sees.bangor.ac.uk. Dr Latif established his BSc and MSc from the Isfahan university of technology and Tehran Polytechnique in Iran, respectively, and PhD from the university of Sheffield, UK. He is currently with the Telematics department of the Open University, Milton Keynes. His research interests include software fault tolerance, dependability aspects of safety-critical systems, using expert systems for dependability assessment, and distributed computer control systems. He has published over 35 papers in national and international conferences and journals.

Dr S. Bennett, Department of Automatic Control and Systems Engineering, The university of Sheffield, Mappin street, Sheffield, S1 3JD, UK. Internet (e-mail): S.Bennett@sheffield.ac.uk. Dr. Stuart is Reader in Automatic Control and Systems Engineering at the University of Sheffield. His current research interests include the design of dependable systems including the incorporation of smart actuators and voting systems. As part of this work he is involved in reliability modelling of mixed hardware/software systems.

Dr J. M. Bass, Merit International, UK. Internet (e-mail): jbass@merit-int.com. Dr Bass is currently a Lead Training and Development Consultant with Merit International working on enterprise customer relationship management systems for major international companies. Previously he was a lecturer in computer systems engineering at the School of Informatics, University of Wales, Bangor. His research interests include real-time software engineering and dependable software systems. He obtained a BSc (Hons) in Computer Systems Engineering from the University of Wales, Bangor and PhD from the University of Sheffield, UK. Dr Bass is a Chartered Engineer and is a member of the BCS, ACM and IEEE.