Adaptive Majority Voter: A Novel Voting Algorithm for Real-Time Fault-Tolerant Control Systems

G. Latif-Shabgahi, and S. Bennett

Department of Automatic Control and System Engineering
The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK
Tel: +44 114 222 5657, Fax: +44 114 273 1729
Contact author: g.latif@sheffield.ac.uk

Abstract

Voting algorithms have been widely used in the realisation of fault-tolerant systems. We introduce a new algorithm of software voting termed as adaptive majority voter. It uses the history of modules of a N-Modular Redundant (NMR) system to select the result of the most reliable module if it contributes toward a majority consensus. Furthermore, a new method for on-line creation of a history record of modules in a Triple Modular Redundant, TMR, system is proposed. The history vector of modules are not only used to improve the behaviour of a wide range of traditional voters, but also can be used to identify the most erroneous/faulty modules of a system to take appropriate reconfiguration or damage preventive strategies. The empirical results show that the novel 'adaptive majority voter' has higher safety and availability levels than the traditional majority voter.

1. Introduction

Redundancy has been widely used to increase the fault-tolerance of physical systems. A fault masking system uses redundant modules and a voting algorithm to mask the faulty/erroneous module outputs. The approach prevents the propagation of wrong results to the subsequent parts of a system and therefore increases the system safety. Many voting strategies have been defined in the literature, see [1], [2], [3] and [4], from which the majority voter and its modified versions have been widely used. The majority voter with n inputs produces an output if an agreement exists between at least $[(n+1)/2]$ of module results. In this case one of the agreed results is arbitrarily selected as the voter output. In cases of disagreement, the majority voter produces an exception code which leads the system toward a fail-safe state (in safety-critical systems) or a fail-stop state (in fail-silent systems). Therefore, the majority voter in a TMR system masks the failure of any single redundant module in each voting cycle.

However, in many applications identifying faulty module(s) after masking is necessary to obtain an acceptable level of system availability. Depending on the nature of application, by identifying faulty channel(s) the corresponding module can be replaced on-line with a (one of the) back-up module (if exists) or is disconnected from the system to contribute to the voter output or is shut-down in a safe manner. These strategies are referred as '*system reconfiguration*', '*system degradation*' and '*fail-stop*' in the context of fault tolerant systems [5]. One approach to identify faulty modules is the use of their history record of within the voting algorithm. In this method a module with the worst history record is taken as the most erroneous/faulty module during the system operation. The history record of modules can also be used to improve the performance of voting algorithm. There is a little published work on the use of module history records in voting algorithms.

In [6] the cumulative number of times each version (software module of a TMR system) has participated in a majority agreement (exact, bit-by-bit majority) is taken as a history measure of versions. In cases of disagreement between version results, the output of a version with the highest history count is chosen as the voter output. The paper showed that such a voter has lower failure probability (defined as the ratio of failed computations to total computations) over the standard majority voter. The three-domain predictor voter [3] uses the fault record of variants in producing voter output in cases of disagreement. In this algorithm, a number f_i is associated with each variant which denotes the number of cycles in which the variant has not contributed toward a majority consensus by that cycle. The number f_i is incremented each time a variant result does not participate in the consensus. In cases of complete disagreement, the closest variant result to the estimated output value with a distance within a predefined predicted threshold is selected as the voter output if its associated fault number is lower than the average fault record value (at any time, the average fault record value is the mathematical average of variants fault number which has been counted by that time). If the associated fault number of selected variant result is less than the average fault record value, an exception flag is generated, that is, the variant is interpreted as a low-reliable module.

Both of the mentioned research works use the 'recent history' of modules in disagreement cases to force an output to occur. In this paper we show that the use of history record of modules can also improve the reliability of a TMR system in majority consensus cases, the issue which has not been addressed in the literature. It will be indicated that a majority voter in which the output selection mechanism [7] uses the recent history of modules in agreement voting cycles, produces more correct and less incorrect results than the standard majority voter during *n* runs.

The paper has been organised as follows. In section two a formal method for providing the history record of modules is defined. In section three the benefits of history records in a triple modular redundant system are discussed. Section four introduces the novel '*adaptive majority voter'* which uses the history record of modules to produce output. Experimental methodology and test results of the adaptive majority voter against the standard majority voter are presented in section 5. Some conclusions are given in section 6.

2. Building the history record of modules

Consider a Triple Modular Redundant system with an inexact majority voter with threshold *a* as shown in Figure 1.

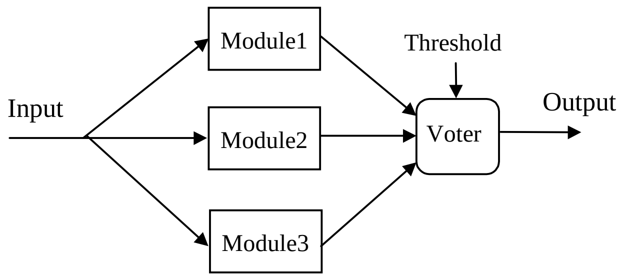

Figure 1. A TMR system with majority voter

One way to build a history for a module m_i is that the cumulative number of cycles in which the module result is contributed toward a majority consensus is computed; then, at any given time a module with the largest cumulative number is interpreted as the most reliable and the module with the smallest number is taken as the lowest reliable module. For a NMR system, the strategy is defined formally as follows:

1. Examine the following relation for any voter input *i* at each voting cycle:

$$|x_i - x_j| < a \text{ where } i, j = 1,2,3,\dots N \text{ and } i \neq j$$

If the inequality is correct, set the parameter a_{ij} to 1, otherwise set a_{ij} to 0. Therefore, $a_{ij} = 1$ means that there is an agreement between the module outputs *i* and *j* at that voting cycle.

2. Build the following set of *(N-1)* Boolean values for each module *i*,

$$\{a_{i2} \ , \ a_{i3} \ , \ a_{i4} \ , \ \dots a_{iN}\}$$

if at least *(N-1)/2* of the values of this set is 1, set the parameter S_i to *1*, otherwise choose $S_i = 0$. Therefore, $S_i = 1$ indicates that the module *i* has been contributed to majority consensus with the other modules in that voting cycle.

3. Accumulate the S_i values of consecutive cycles of channel *i* during *n* system runs to obtain $H_i(n) = \sum_{l=1}^{n} S_i(l)$. This value indicates the total number of contributions of module *i* toward a majority consensus during *n* runs. It is, in fact, the history record of module *i*.

4. The normalised value $P_i = \frac{1}{n}.\sum_{l=1}^{n} S_i(l)$ is, then, a criterion for the reliability level of module *i*. A module with the highest P_i is, in fact, the most likely correctly performing module.

Moreover, at any given cycle *q*, the value $P_i(q) = \frac{1}{q}.\sum_{l=1}^{q} S_i(l)$ indicates the state of module *i* up to that cycle.. This value is called '*state indicator'* of module *i* at cycle *q*. The value $H_i(q) = \sum_{l=1}^{q} S_i(l)$ is referred as '*history record'* [or contribution toward majority consensus record] of module *i* up to the cycle *q*.

3. Using the history record of modules

The history record of modules can be used in different ways in a TMR system:

1. Off-line using of history record
   - to identify the most faulty/erroneous module during the system operation time
2. On-line using of history records
   - to reconfigure the system structure, degrade, fail-stop or safe-fail

- to improve the performance of voting algorithm during operation time

The second issue is the interest of this paper. However, in the following, we briefly explain the benefits of the first issue.

3.1. Off-line identification of faulty modules

In this case, the identification of faulty module(s) is carried out at the end of system operation time. When a system terminates its operation, normally or in on emergency, an identification procedure processes the final P_i values of modules to recognise the most problematic module(s). Such a module is, then, repaired or replaced with a non-faulty back-up one for next mission. Since in this method the system is not monitored during its operation time, the user has to be ensure that the system safety and availability are not threatened during its mission time, even in the presence of faults. The method, therefore, is convenient for TMR system in which only one of the modules is likely to be faulty whereas the other two have high level of reliability. It is unsuitable for TMR systems in which more than one module is likely to be fault prone. Where more than one module produces the wrong result, the voter generates '*no-result*' and persistence of such condition makes the system cease its operation. This system may provide a low level of availability since early termination of system operation, in most applications such as aeroplane, is not acceptable. It is required that the system continues its operation even in the presence of two faulty modules as long time as possible.

3.2. On-line processing of history records

The history record of modules may be used, on-line, in two different ways:

3.2.1. In the first method, the state indicator of modules are continuously monitored but are used periodically (every n_h voting cycles) during the system operation time. The length of period is application specific. The set of monitored results which is indicated by $P = \{P_1 , P_2 , P_3\}$ are, then, used to take appropriate strategies to improve the system operation during the subsequent period. The strategy may require *i)* to disconnect the most faulty/erroneous module during the recent period and reconfigure or degrade the level of system redundancy, *ii)* to change the parameters of voter (e.g., change the voter thresholds, change the output selection mechanism of voter, adjust weighting factors, etc.). Of these, the second issue is the interest of this paper. To clarify consider a three input majority voter with randomly output selection mechanism as defined in [1] Such a voter, in cases of agreement between redundant results selects randomly one of the agreed values as voter output. The selection may be incorrect choice in some cases. For example, consider a voting cycle with notional correct value 10 in which the module results are: $(x_1 = 10 , x_2 = 10.4 , x_3 = 10.7)$. By considering the threshold value equal to 0.5, an agreement exists between x_1 and x_2 and also x_2 and x_3. Now, if x_3 is randomly selected as the voter output, the choice is, in fact, a wrong decision. The problem can be solved by using history record/state indicator of modules as the basis of voter output selection mechanism. A majority voter which uses the periodical history record of modules, selects the most reliable result among the agreed values. For example, if the latest state indicator of modules is represented by: $P = \{P_1 = 0.87 , P_2 = 0.81 , P_3 = 0.49)$, then one can be ensured that with result set $(x_1 = 10 , x_2 = 10.4 , x_3 = 10.7)$ the value x_3 is never selected as voter output, in spite of that it is one of the agreed results of the present voting cycle. The reason is that x_3 belongs to the least reliable module (with the least value of P_i). Note that $P_i \subset [0 \;\; 1]$ and a result which is produced by a module with $P_i \to 1$ is the most reliable output and a result supported by $P_i \to 0$ is the least reliable output.

In this selection, we implicitly introduced a new version of inexact majority voter, a version which selects the most likelihood reliable result as its output in agreement cases. There is a fundamental problem with this method. A module which produces low-reliable results in a specific period of time *(e.g.,* during the first *1000* voting cycles) may generate high reliable outputs in the subsequent period. The reason is that environmental conditions and transient faults or a particular trajectory of input values which make a specific module to produce low reliable results for the period T_i may be removed or changed during the subsequent period T_{i+1}. Therefore, the construction of the output selection mechanism of a voter based on the history record of modules in the previous period may lead to loose some reliable results in the present period. However, this method is effective when the modules have drift faults. A module which has a drift during the period T_i, has more likely the same problem in the period T_{i+1} and therefore can be monitored by periodical history records, without having the aforementioned problem.

3.2.2. In the second method, the state indicator of modules which are continuously computed in each run, are used in the subsequent cycle to produce (select or compute) a more reliable output and/or inform the management system about the on-line status of modules. For example, if at the end of cycle *90* the state indicator of modules is represented by the set: $P = \{P_1 = 0.87 , P_2 = 0.81 , P_3 = 0.49)$, in cycle *91* among the results $(x_1 = 10 , x_2 = 10.4 , x_3 = 10.7)$ the value x_3 which is contributed to agreement, is not selected as the voter output. This is the basic idea of adaptive majority voter which will be formally introduced in the next section. The creation and use of state indicator within the voting algorithms leads to introduce a class of novel versions of existing voters.

4. Adaptive Majority voter

This voter, in cases of agreement between redundant module results, selects the output of the most reliable module; the module which has had the largest number of contribution toward a majority consensus so far. The voter is defined formally as follows:

1. Let $A = \{x_1 , x_2 , x_3 , \ldots, x_N\}$ denotes the set of N module results at voting cycle q, and $H = \{H_1 , H_2 , H_3 , \ldots, H_N\}$ is the history record of modules up to cycle q. Construct a partition $V_1, V_2, \ldots, V_k$ of A where for each i the set V_i is maximal with respect to the property that for any x and y in V_i, $d(x, y) \leq a$. where a is the voter threshold.

2. Let V be the set in the partition $V_1, , V_k$ of the largest cardinality.

3. If $|V| \geq (N + 1)/2$, then select the value from V as voter output which has the largest history record value, where $|V|$ denotes the cardinality of the set V.
3a. For all x_j in partition V, increment history record whereas for other x values do not change the corresponding H values. A new history record, then, is constructed which is the base of output selection mechanism of voter in the next voting cycle.

4. If $|V| < (N + 1)/2$, a '*no-result*' output is produced as the voter output and no history record is changed.
5. At the end of operation time (after n voting cycle) the state indicator set:

$$P = \{p_1 , p_2 , p_3 ,\ldots, p_N\} \text{ where } P_i = \frac{1}{n}.\sum_{l=1}^{n} S_i(l)$$

indicates the reliability level of modules.

Example. The inputs of a five input adaptive majority voter in a particular system cycle q, are as follows: $A = \{10 , 10.2 , 11.5 , 11.2 , 10.5\}$. The history record of modules up to cycle q is represented by: $H_{q-1} = \{96 , 94 , 82 , 79 , 89\}$. Determine the voter output and the new history record at the end of this cycle.

$V_1 = \{10 , 10.2 , 10.5\}$ $\quad |V_1| = 3$
$V_2 = \{11.2 , 11.5\}$ $\quad |V_2| = 2$
Since $|V_1| \geq (5+1)/2$, then $V = V_1$

The first component of V, $x = 10$, has the largest state value, therefore the voter output is 10.

$\{a_{12} , a_{13} , a_{14} , a_{15}\} = \{1 , 0 , 0 , 1\} \rightarrow S_1 = 1$
$\{a_{21} , a_{23} , a_{24} , a_{25}\} = \{1 , 0 , 0 , 1\} \rightarrow S_2 = 1$
$\{a_{31} , a_{32} , a_{34} , a_{35}\} = \{0 , 0 , 1 , 0\} \rightarrow S_3 = 0$
$\{a_{41} , a_{42} , a_{43} , a_{45}\} = \{0 , 0 , 1 , 0\} \rightarrow S_4 = 0$
$\{a_{51} , a_{52} , a_{53} , a_{54}\} = \{1 , 1 , 0 , 0\} \rightarrow S_5 = 1$

The new history record is : $H_o = \{97 , 95 , 82 , 79 , 90\}$.

5. Experimental methodology and results

To compare the behaviour of adaptive majority and standard majority voters in coping with perturbed inputs, a set of experiments is carried out. Each voter was placed in a TMR configuration of redundant inputs in which the consecutive inputs with sinusoidal profile; $100 + 100.sin(t)$; are perturbed via intermediate saboteurs. The voter threshold is assumed 0.5 and the input sample-rate is taken 0.1. Each saboteur injects a random error with uniform distribution from interval $[-\theta \quad +\theta]$ into any sampled input in each voting cycle. The maximum amplitude of error which is injected by saboteurs can be equal or different. For example, in a particular experiment (each experiment set includes $10,000$ runs) we assume that the first and second saboteurs perturb input data with errors from interval $[-1 \quad +1]$ whereas the third saboteur injects errors from interval $[-3 \quad +3]$. In other words, it is assumed that the third module of the TMR system is more error prone than the other two.

In each experiment set, having performed $10,000$ runs, the number of correct (n_c) results, incorrect results (n_{ic}) and disagreed results are collected. The measure $S = 1 - \frac{n_{ic}}{n}$ is defined as the safety index and the value $A = \frac{n_c}{n}$ is taken as the availability index of voters. Ideally, $S = 1$ and $A = 1$. This means that a voter with measure $S = 1$ is a complete safe voter. Since the notional correct answer is known for each cycle, where a voter output (the value with the majority consensus) diverges from the notional correct answer with a value less than 0.5 (this is the accuracy threshold of the system and has been taken, here, equal to the voter threshold), it is taken as a correct result, otherwise it is assumed an incorrect result. In no majority consensus situations, the voter produces a '*no-result*' output which represents a disagreement case.

The experimental result of voters with different noise levels: $f_1 \subseteq [-1 \quad +1]$, $f_2 \subseteq [-2 \quad +2]$ and $f_3 \subseteq [-2.5 \quad +2.5]$ is as follows:

majority: $(n_c = 2236 \ , \ n_{ic} = 3139 \ , \ n_d = 4625)$
adaptive majority: $(n_c = 2419 \ , \ n_{ic} = 2956 \ , \ n_d = 4625)$
and module reliability values after all runs are:
$(p_1 = 0.4012 \ , \ p_2 = 0.3965 \ , \ p_3 = 0.3532)$

The results show that the adaptive majority voter gives more correct and less incorrect results than the standard majority voter *(1.8%)* while their number of disagreed outputs are the same, as expected. The improvement value, *1.8%,* in spite of being small, is considerable in safety critical systems. The *module's reliability set* also shows that in this error scenario, the first module is the most reliable and the last module is the least reliable one as expected according to the assumption. Since in reality we do not know more about the effective noise/error level of modules, creation and the use of module's reliability set will play an important role in voter output selection process and system diagnosing.

Another experimental results with equal (maximum) but large noise levels: f_1 , f_2 , $f_3 \subseteq$ [-3 +3] are as follows:

majority: $(n_c = 738\ ,\ n_{ic} = 3463\ ,\ n_d = 5799)$
adaptive majority: $(n_c = 760\ ,\ n_{ic} = 3441\ ,\ n_d = 5799)$
and module reliability values after all runs are:
$(p_1 = 0.2934\ ,\ p_2 = 0.2953\ ,\ p_3 = 0.29)$

The results show that with high amplitude noise/errors, there is not a considerable performance difference between the voters. The reason is that both of the voters, in this case, have a high probability of reaching disagreement (this probability, for example, here is about 58%) as well as wrong agreement (here is about 34%).
The third experimental results reported in the following are shown that with equal (maximum) but small channel noise levels; f_1 , f_2 , $f_3 \subseteq$ [-0.7 +0.7]; the adaptive majority voter has a considerable (about 12.5%) performance improvement over the standard majority voter:

majority: $(n_c = 7410\ ,\ n_{ic} = 2356\ ,\ n_d = 234)$
adaptive majority: $(n_c = 8669\ ,\ n_{ic} = 1097\ ,\ n_d = 234)$
and module reliability values after all runs are:
$(p_1 = 0.8138\ ,\ p_2 = 0.8152\ ,\ p_3 = 0.8165)$

The final results are obtained with equal maximum error amplitude at the first and second voter inputs, f_1 , $f_2 \subseteq$ [-1 +1], and a higher error amplitude at the third input, f_3 = [-2 +2].

majority: $(n_c = 3708\ ,\ n_{ic} = 3233\ ,\ n_d = 3059)$
adaptive majority: $(n_c = 4111\ ,\ n_{ic} = 2830\ ,\ n_d = 3059)$
and channel reliability values after all runs are:
$(p_1 = 0.5776\ ,\ p_2 = 0.574\ ,\ p_3 = 0.3926)$
In this case the adaptive majority voter has only 4% better performance than the majority voter.

All of the above test results show that the adaptive majority voter is superior to the standard majority voter A wide range of tests with other error scenarios shows that when two or three channels have equal maximum noise/error amplitude and errors are small in amplitude, adaptive majority voter gives better (more safe and more available) results. To indicate this issue more clearly, the behaviour of voters is compared graphically in other 9 error scenarios. In each scenario which includes 10^4 runs, the maximum amplitude of error which is injected by the first and second saboteurs are equal whereas the third saboteur injects errors with higher amplitude. 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 denotes the allowed maximum amplitude of errors being injected to the first voter input, the second element of the tuple represents the allowed maximum amplitude of errors being injected to the second voter input, etc. Each tuple is called an *error tuple.* Figure 2 shows the availability of voters versus error tuples.

Figure 3 indicates the safety plot of voters versus error tuple. For readability, only the first element of each error tuple has been indicated on the figures as x-axis labels.

6. Conclusions

The paper addressed the benefits of considering 'history record' of modules within the standard majority voter in a triple modular redundant system. A formal method for creating the history record of modules in a NMR system was defined and a new software voting algorithm, *adaptive majority voter,* was introduced. The novel voter uses the history record of modules to identify the result of the most reliable module at those voting cycles in which a majority consensus is reached. It behaves as the standard majority voter in disagreement cases. The experimental results demonstrated the superiority of the novel voter against the standard majority voter in terms of safety and availability in different error scenarios.

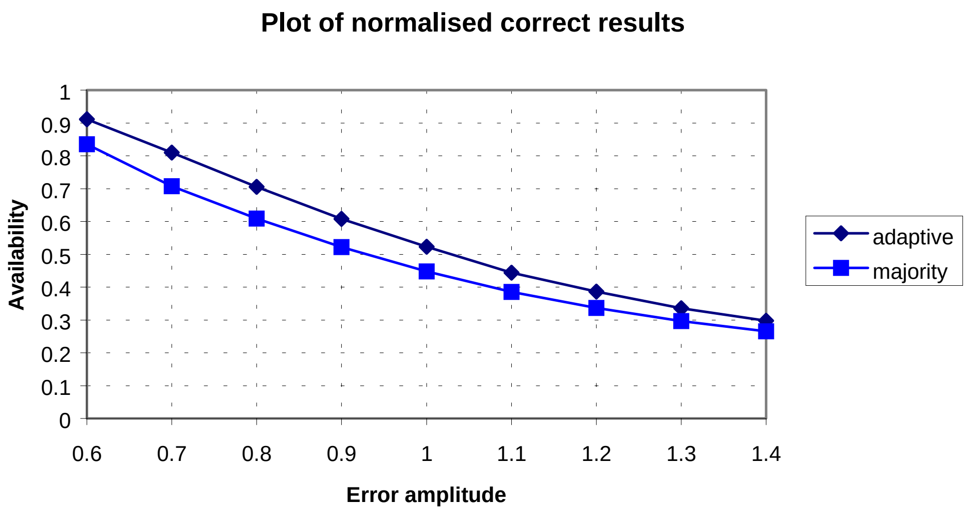

Figure 2. Comparative availability results of basic majority and novel adaptive majority voters

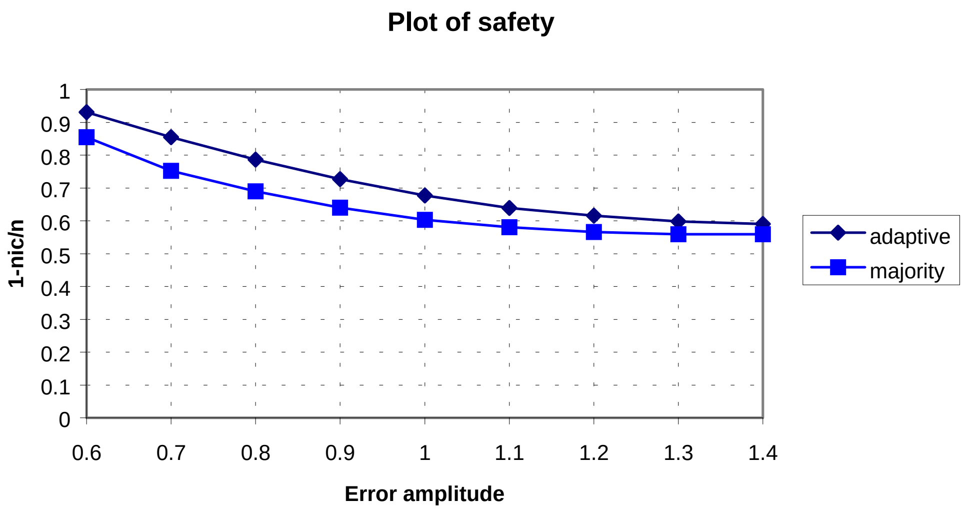

Figure3. Comparative safety results of basic majority and novel adaptive majority voters

7. References

[1] Lorczak, P. R., Caglayan, A. K., and Eckhardt, D. E.(1989). "A Theoretical Investigation of Generalised Voters", Digest of papers, IEEE 19th Int. Ann. Symp. on Fault- Tolerant Computing Systems, Chicago, June 1989, pp. 444-451.

[2] **Kanekawa, K.,** Maejima, H., Kato, H., and Ihara, H. (1989). "Dependable On-Board Computer Systems with a New Method: Stepwise Negotiated Voting", Digest of papers IEEE 19th Int. Ann. Symp. on Fault-Tolerant Computing Systems, Chicago, June 1989, pp. 13-19.

[3] Bass, J. M. (1995). "Voting in Real-Time Distributed Computer Control Systems", PhD thesis, Department of Automatic Control and System Engineering, The University of Sheffield, Sheffield, UK.

[4] Latif-Shabgahi, G., Bass, J. M., and Bennett, S. (1998a). "Complete Disagreement in Redundant Real-Time Control Applications", Digest of papers, 5th IFAC Workshop on Algorithms and Architectures for Real-Time Control, AARTC'98, Cancun, Mexico, April 1998, pp. 259-264.

[5] Johnson, B. W. (1989). "Design and Analysis of Fault-Tolerant Digital Systems", Addison-Wesley Publishing Company

[6] Gersting, J. L., Nist, R. L., Roberts, D. B., and Van Valkenburg, R. L. (1991). “A Comparison of Voting Algorithms for N-Version Programming”, Digest of papers IEEE 24th Ann. Hawaii Int. Conf. on Systems Sciences, Vol. 2, pp. 253-62.

[7] Latif-Shabgahi, G., Bass, J. M., and Bennett, S. (1998b). “Component-Oriented Voter Model for Dependable Control Applications”, Proc. of the Int. Conference on Control’98, University of Wales, Swansea, UK, Sept. 1998,