On Voting Strategies for Loosely Synchronized Dependable Real-Time Systems

Hüseyin Aysan[1], Radu Dobrin[1], Sasikumar Punnekkat[1], and Iain Bate[1,2]
[1]Mälardalen University, Västerås, Sweden
[2]University of York, York, UK
{huseyin.aysan, radu.dobrin, sasikumar.punnekkat}@mdh.se, iain.bate@cs.york.ac.uk

Abstract

Hard real-time applications typically have to satisfy high dependability requirements in terms of fault tolerance in both the value and the time domains. Loosely synchronized real-time systems, which represent many of the systems that are developed, make any form of voting difficult as each replica may provide different outputs independent of whether there has been an error or not. This can also lead to false positives and false negatives which makes achieving fault tolerance, and hence dependability, difficult. We have earlier proposed a majority voting technique, "Voting on Time and Value" (VTV) that explicitly considers combinations of value and timing errors, targeting loosely-synchronised systems. In this paper, we extend VTV to enable voter parameter tuning to obtain the desired user specified trade-offs between the false positive and false negative rates in the voter outputs. We evaluate the performance of VTV against Compare Majority Voting (CMV), which is a known voting approach applicable in similar contexts, through extensive simulation studies. The results clearly demonstrate that VTV outperforms CMV in all scenarios with lower false negative rates.

1 Introduction

Safety critical real-time systems typically have to guarantee highly dependable performance due to their interactions with, and possible impacts on, the environment. Satisfying dependability requirements of such systems typically involves fault prevention, fault tolerance as well as fault forecasting approaches in their design. In this paper we focus on fault tolerant design by the use of redundancy, which is a widely used approach in safety critical systems, mainly in the form of the well known N-modular redundancy (NMR) paradigm. Due to, e.g., space and cost constraints, in practice this paradigm is typically used in its basic configuration, i.e., triple-modular redundancy (TMR) where three nodes are used for replication [18] and one error at a time can be masked in any node. The key attraction of this approach lies in its low overhead and fault masking abilities, without the need for backward recovery [14]. Some of the disadvantages include the cost of redundancy and that the voter is a possible single point of failure. Traditionally, voters are constructed as simple electronic circuits so that a very high reliability can be achieved. Distributed voters have also been employed to take care of the single-point failure mode in case of highly critical systems [8, 17]. With the additional cost of increased computation time, more enhanced voting strategies, such as *plurality*, *median* and *average* voters, can be performed in software. *Plurality* voters (or *m-out-of-n* voters) require m corresponding outputs out of n, where m is less than the majority, to reach a consensus [16, 10]. *Median* voters output the middle and *average* voters output the average value of the replica output values. Surveys and taxonomies for voting strategies have been presented [3, 9, 15].

An issue related to value voting is that the nodes' output values can vary slightly due to e.g., hardware imprecision, resulting in a range (or a set) of values which should be considered as correct in order to avoid problems indicated in [5, 6]. In order to accomplish this, *inexact voting strategies* have been proposed [13, 20, 23]. This phenomenon is also observed in the time domain due to several factors, such as clock drifts, node failures, processing and scheduling variations at node level, as well as communication delays. Most of the existing voting strategies, however, focus solely on tolerating anomalies in the value domain by assuming that they are running on *tightly synchronized systems*, as presented in [11]. On the other hand, using *loosely synchronized systems* is an attractive, often used, alternative due to the lower overheads, reduced complexity, and the lower reliance on the synchronization mechanism itself [12]. The key problem with loosely synchronized systems is that voting is made more complicated as differences between the replicas' values will exist independent of whether errors have occurred as each of the replicas will be receiving different inputs to its calculations. This can also lead to false positives and false negatives which makes achieving

fault tolerance, and hence dependability, difficult.

A simple approach towards tolerating both value and timing errors using the NMR approach could be adding time stamps to the replica outputs. Then, voting on the time stamps could detect possible timing errors in the replica outputs. However, this approach is unable to mask late timing errors since the voter has to wait for all the values to be delivered by the replicas. Majority voting techniques that are able to implicitly handle the errors in the time domain, have been proposed by Ravindran et al., [21, 22], and Shin et al., [24]. In these approaches, voting is performed among a quorum or a majority of responses received, rather than waiting for all the responses, in order to be able to mask late timing errors. Both Quorum Majority Voting (QMV) and Compare Majority Voting (CMV) provide outputs within a bounded time interval. These approaches make an assumption that the number of timing errors within a certain period does not exceed an allowed threshold. As long as the assumption holds, the ability to detect value errors is equivalent to existing approaches. QMV and CMV cannot detect whether this *assumption violation* has occurred. Hence, these approaches may produce optimistic results in the sense that, e.g., an incorrect value may be produced at a correct (or incorrect) time, and thus may not be fully suitable for hard real-time systems.

Traditionally, research efforts are less focused on scenarios and solutions beyond the stated assumptions, whereas in practice, the robustness of dependable systems can be enhanced by the provision of signalling assumption violations. For example, in the event of a violation of the underlying assumptions, a voter needs to be cautious in the provision of outputs to the environment, e.g., a "no output" together with an error signal may be a better alternative than a potentially erroneous output.

In [2] we have outlined a conceptual design for a real-time voting strategy Voting on Time and Value (VTV), which performs voting in both the time and the value domains. In particular, VTV aims to enhance the fault tolerance abilities of NMR by ensuring the output from the voter to be both correct in value, and delivered within a specified admissible time interval, under specified assumptions. VTV is designed to detect when the assumption violation made by QMV and CMV is broken. VTV still has an upper bound on the number of errors that can be handled, however the limit is improved. This gives an enhanced capability for ensuring fail-safe or fail-stop behavior within systems. VTV is also designed to perform no worse than existing strategies for systems that are tightly synchronised where the effect of the assumption violations do not exist. However, the earlier results in [2] were not supported by any empirical evaluation regarding the performance of VTV in relation to other similar approaches. Another important aspect left unexplored in the previous work, is the sensitivity of the voter outputs on the detection thresholds. An inappropriate selection of the thresholds could either make the results overly-cautious or could result in erroneous outputs being accepted by the voter violating the system requirements.

The contributions of the paper are as follows:

1. extension of the technique in [2] so that the detection threshold is made tunable and the exploration of the effect of the tuning
2. an evaluation that shows how different error sizes and error scenarios affect the detection capabilities. The results demonstrate the robustness of VTV, confirming that VTV outperforms CMV in terms of its potential for notifying assumption violations

The rest of the paper is organized as follows: In Section 2 we introduce the system and the error model followed by an overview of the voting strategies designed for loosely synchronized real-time systems in Section 3. In Section 4 we present the evaluation results and we conclude the paper in Section 5.

2 System and Error Model

In this paper, we assume a distributed real-time system, where each critical node is replicated for fault tolerance, and replica outputs are voted on to ensure correctness in both value and time. Upon receiving identical requests or inputs, replicas of a node start their executions on dedicated processors whose clocks are allowed to drift from each other at most by a maximum deviation. This bound can be achieved by relatively inexpensive clock synchronization algorithms implemented in software (compared to expensive tight clock synchronization implementations). After the replicas complete their executions, the outputs are sent to a stand-alone voting mechanism. Deviation in message transfer times from the replicas to the voter is also bounded by using reliable communication techniques. Upon receiving deliveries from replicas, the voter starts executing the voting algorithm and outputs a correct value at an admissible time or signals the non-existence of a correct output to the subsequent component in the system in a timely manner.

We use the following notations for the system properties:

- δ maximum deviation in time domain between any two replica outputs, in an error-free scenario, as perceived by the voter, which includes the maximum skew between any two non-faulty replica clocks δ_{clock} and the maximum skew between any two message transmissions from replicas to the voter δ_{comm}
- C^{voter} worst-case computation time of the voting algorithm
- Δ maximum admissible deviation between any two voter outputs in the time domain relative to the correct time point, t^*, (seen by a perfect observer), as per the real-

time and dependability specifications

α detector coefficient used as part of error detection. In [2] this was based purely on the values of δ and Δ, however in this paper it is made a tunable value

σ maximum admissible deviation in the value domain between any two replica outputs

The reader should note that the maximum admissible deviation between any two voter outputs in the time domain, Δ, relative to the correct time point, t^*, is determined according to the system specifications, i.e., what the rest of the system can tolerate as per the real-time and dependability specifications. On the other hand, δ determines the maximum deviation between any two replica outputs in an error free scenario assured by the implemented clock synchronization procedures. The reader should also note that the maximum admissible deviation of a voter output from a correct time point, t^*, is $\Delta/2$ and the maximum deviation of a replica output from t^* in an error-free scenario is $\delta/2$.

Error detectors use δ multiplied by the detector coefficient α to identify any potential variations in the time domain that may affect the system timeliness. If α is less than one, the error detector may identify even error-free outputs as erroneous increasing the false positives, however this may increase the detection of the variations in the time domain due to errors reducing the false negatives. A value less than one can also be used if $\Delta/2$ is less than $\delta/2+C^{voter}$ to detect any scenarios that may result due to inadequate synchronization and threaten system's timeliness requirements. On the other hand if $\Delta/2$ is greater than $\delta/2 + C^{voter}$, then the system can tolerate even some of the variations due to errors, hence the difference between $\Delta/2$ and $\delta/2 + C^{voter}$ can be used to reduce false positives by using an α value greater than one. Figure 1 shows the relation between Δ, δ and C^{voter}.

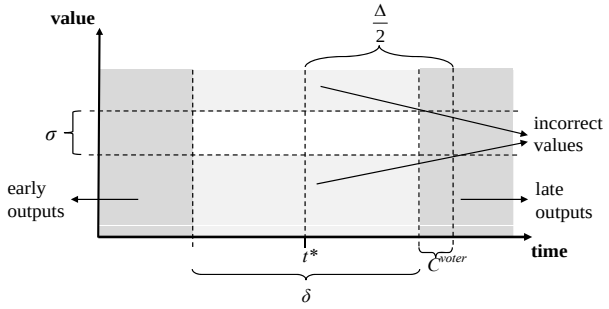

Figure 1. Output correctness in the time and the value domains and the relation between Δ, δ, and C^{voter}

We follow the dependability concepts originally introduced in [1, 4, 19]. For the sake of readability, we denote the i^{th} replica of a given node by R_i. The output delivered by R_i, is specified by two domain parameters, viz., value and time:

$$\textit{Specified output for } R_i = < v_i^*, t_i^* >$$

where v_i^* is the correct value, t_i^* is the correct output delivery time (seen by a perfect observer).

An output delivered by R_i is denoted as:

$$\textit{Delivered output from } R_i = < v_i, t_i >$$

where v_i is the value and t_i is the time point at which the value was delivered. Based on the voter parameters σ, δ and α, we define the output generated by replica R_i as *incorrect in value domain* if:

$$v_i < v_i^* - \frac{\sigma}{2} \text{ or } v_i > v_i^* + \frac{\sigma}{2}$$

and *incorrect in time domain* if:

$$t_i < t_i^* - \alpha\frac{\delta}{2}\ (early\ timing\ error)$$

or if

$$t_i > t_i^* + \alpha\frac{\delta}{2}\ (late\ timing\ error).$$

The notations used for the error behavior of the replicas (seen by a perfect observer) are:

- E_v: the number of replicas that have only value errors
- E_t: the number of replicas that have only timing errors, consisting of two subcategories:
  - E_t^e: the number of replicas that produce early outputs with correct values
  - E_t^l: the number of replicas that produce late outputs with correct values
- E_{vt}: the number of replicas that have both value and timing errors, consisting of two subcategories:
  - E_{vt}^e: the number of replicas that produce early outputs with incorrect values
  - E_{vt}^l: the number of replicas that produce late outputs with incorrect values

where the total number of erroneous replica outputs E is:

$$E = E_t + E_v + E_{vt}$$

Finally, the notations used for describing the voting mechanisms are:

- N: number of replicas
- M_t: minimum number of replicas required to form a consensus in time domain, as per system specification
- M_v: minimum number of replicas required to form a consensus in value domain, as per system specification

Basic assumptions: The voting approaches presented in this paper rely on the following set of basic assumptions (to a large extent based on [7]):

A1: non-faulty nodes produce values within a specified admissible range after each computation block

A2: non-faulty nodes produce values within a specified admissible time interval after each computation block

A3: replica outputs with incorrect values do not form (or contribute in forming) a consensus in value domain

A4: incorrectly timed replica outputs do not form (or contribute in forming) a consensus in time domain

A5: there exist adequate mechanisms, e.g., infrequent synchronization, which are significantly less costly than tight synchronization, to ensure a maximum permissible replica deviation from the global time

A6: the voting mechanism does not fail, as it has been designed and implemented as a highly reliable unit

3 Voting Strategies for Loosely Synchronized Systems

Table 1 presents an overview of various voting strategies, applicable to loosely synchronized real-time systems which are described in the following sections.

3.1 Compare/Quorum Majority Voting (CMV/QMV)

Shin et al. presented two voting techniques [24], viz., *Quorum Majority Voting (QMV)* and *Compare Majority Voting (CMV)* that relax the tight synchronization requirements. QMV performs majority voting among the received values as soon as *2n+1* out of *3n+1* replicas deliver their outputs to the voter, thus, guaranteeing detection of majority of non-faulty values even in the case *n* replicas produce erroneous outputs. CMV masks *n* erroneous outputs out of *2n+1* replica outputs as in basic majority voting. The main difference is that in CMV, the voter output is delivered as soon as a majority consisting of identical values has been received, without waiting for the rest of the replicas. Both QMV and CMV provide outputs within a bounded time interval, as long as the assumptions regarding the maximum number of errors hold. However, QMV and CMV are unable to detect any assumption violations in the time domain.

3.2 Voting on Time and Value (VTV)

Previously in [2], we presented a voting strategy that explicitly considers errors in both value and time domains. The correctness of the approach relies on a number of conditions regarding the permissible number of replicas that produce erroneous outputs:

$\mathcal{C}1$: The number of replicas that produce erroneous outputs can not exceed the difference between the total number of replicas and the minimum number of error-free replicas required to achieve consensus in the value domain.

$$E_v + E_t + E_{vt} \leq N - M_v$$

$\mathcal{C}2$: The number of replicas that produce erroneous outputs in time domain is bounded by the difference between the total number of replicas and the minimum number of error-free replicas required to achieve consensus in the time domain.

$$E_t + E_{vt} \leq N - M_t$$

The goal of this approach is two fold:

1. always deliver the correct value within $[t^* - \frac{\Delta}{2}, t^* + \frac{\Delta}{2}]$, if the conditions $\mathcal{C}1$ and $\mathcal{C}2$ hold
2. provide information about the violation of the conditions, otherwise.

In VTV, agreement in the time domain is reached when M_t out of N replicas deliver their outputs within the time interval $[t^* - \alpha\delta/2, t^* + \alpha\delta/2]$ (referred to as *feasible window* henceforth).

The maximum number of sets, consisting of M_t consecutive replica outputs each (out of the N replicas), is $N - M_t + 1$. Since the consensus in time domain can be reached in any of these sets, a separate feasible window needs to be initiated upon receiving each of the first $N - M_t + 1$ replica outputs. In order to keep track of the feasible windows, simple countdown timers are utilized. Once an agreement in time domain is obtained, then values are voted. If an agreement in the value domain is not obtained within a particular feasible window, the process continues with subsequent feasible windows, until agreement in both the time and the value domains can be achieved, or violations of $\mathcal{C}1$ or $\mathcal{C}2$ are detected.

Depending on the real-time application characteristics, a value produced by a node may be considered *valid* or *invalid* for the purpose of voting, in the case it is produced early. Hence, we have two cases:

Case 1 *Only timely outputs are considered valid.* If a plurality exist among the *timely* received values, the plurality value is delivered as the correct output.

Case 2 *Early and timely outputs are considered valid.* If a plurality exist among *all* the received values, the plurality value is delivered as the correct output. The advantage of this case is that the number of the nodes required to mask a given number of errors (in the time and the value domains) can be significantly reduced,

Voting Strategy	Description	Voting domain(s)
QMV	1. Wait for a quorum (2n+1 out of 3n+1 replica outputs) 2. Perform majority voting among the quorum	value
CMV	Wait for a majority (n+1 out of 2n+1 replica outputs) with identical values	value
VTV (Case 1)	1. Wait for a majority (or plurality) delivered within a predefined time window 2. Perform voting in the value domain among timely replica outputs 3. In case there is no agreement in the value domain, return to step one (or signal disagreement in case it was the last possible plurality in time)	value and time
VTV (Case 2)	1. Wait for a majority (or plurality) delivered within a predefined time window 2. Perform voting in the value domain among available replica outputs 3. In case there is no agreement in the value domain, return to step one (or signal disagreement in case it was the last possible plurality in time)	value and time

Table 1. Overview of voting strategies suitable for real-time systems

compared to Case 1, since replica outputs erroneous in one domain may still be used to reach consensus in the other domain. However, if the early timing errors have the potential to cause system failures, then using VTV in this configuration may be unadvisable. In this case, Condition $C1$ becomes:

$\mathcal{C}1(Case2)$: The number of replicas that produce erroneous outputs, except the outputs with early timing errors, can not exceed the difference between the total number of replicas and the number of error-free replicas required to achieve consensus in value domain.

$$E_v + (E_t - E_t^e) + (E_{vt} - E_{vt}^e) \leq N - M_v$$

4 Evaluation

In this section, we present a simulation study performed in order to investigate the performance of VTV compared to CMV. As shown in Figure 2, we simulated five nodes where various kinds of transient errors were injected with certain probabilities along with a *reference node* that never fails. The outputs of the five nodes were voted using three different voters: (i) one implementing the VTV strategy where early generated replica outputs are considered invalid for the value voting (Case 1), (ii) one implementing the VTV strategy where early outputs are considered valid for the value voting (Case 2) and (iii) one implementing the CMV strategy. All node outputs were sent to a *perfect observer* module together with the voter outputs and the assumption violation signals from the voters, in order to determine the false positive rate (FPR) and the false negative rate (FNR) of the different voting strategies. This section is structured as follows. First, the experiment setup is given, then the voters are compared with $\alpha = 1$, and finally the effect of different detection thresholds, α, is considered.

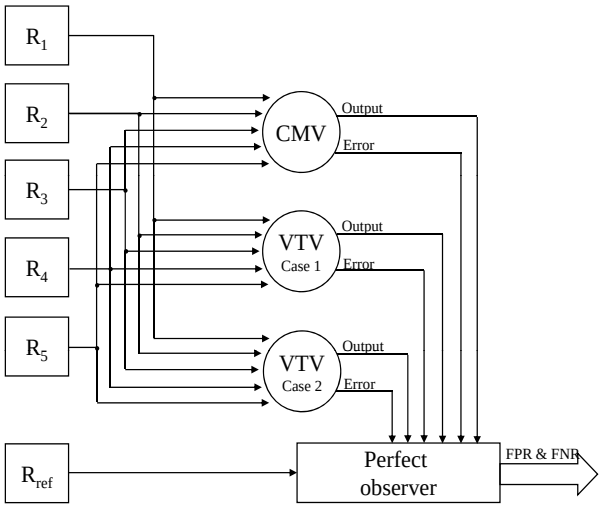

Figure 2. Simulation setup

4.1 Experimental design

The simulations were implemented using Matlab/Simulink. In order to evaluate the dependency on the signals input to the replicas, two different signals, viz., a sine wave and a square wave, at various frequencies were used. By using the sine wave, we simulated signals that change smoothly over time, with a value limit for the maximum change during a given time. The sine wave also allows an assessment of how specific frequencies are affected. An example for such a signal is the reading of a temperature sensor in a closed room, where the change rate is limited based on a number of factors such as the heater capacity, the volume of the room, etc.. A square wave also allows a range of frequencies to be assessed. By using the square wave, we simulated signals such as those that

can be generated by sensors that output boolean values. An example for such a sensor is a smoke detector which outputs either *false* to indicate that there is no smoke, or *true* otherwise. The amplitudes of the signals were set to 5 units. We simulated the sensor noise by adding a normally distributed random value with a variance of 0.2 units to the input signals.

4.1.1 Injected errors

Three types of errors were considered in the simulations.

1. **Value errors:** The node outputs the sampled value with a uniformly distributed random value offset within a given value offset range.
2. **Timing errors:** The node outputs the sampled value with a uniformly distributed random time offset within a given time offset range.
3. **Omission errors:** When an omission error (which is a special type of timing errors) is injected, the node skips outputting the sampled signal value. Hence the output of the node remains unchanged from the last output value.

In order to evaluate the sensitivity of the voting approaches, i.e. the detection capabilities of the voters depending on the errors' magnitude, the time and value offsets were randomly generated within two different ranges. The narrow time offset range was defined as $[-1ms, 1ms]$. Please note that replica outputs are assumed to be timely as long as the time difference between the output delivery times and the delivery time of an ideal replica (with no timing errors and no time drift from the real-time) is less than or equal to $\alpha\delta_{clock}/2$ where $\delta_{clock} = 0.5ms$ and α is selected within an interval of $[0.6, 1.4]$, hence the timing errors generated within this range are harder to detect than the timing errors generated using the wide time offset range $[-4.5ms, 4.5ms]$, which, together with the drift from the real-time, may result in replica output deliveries any time during a period. The narrow value offset range was defined as $[-1, 1]$ units and the wide value offset range is defined as $[-5, 5]$ units. Figure 3 shows a sampled input signal and a node output with injected errors.

We evaluated voters' abilities to work in a wide range of conditions by injecting the errors with different combinations (no errors, only value errors, only timing errors (including omission errors), and both value and timing of errors) as well as by injecting errors with different probabilities. We conducted two sets of experiments, based on the error occurrence probability during a period for each type of error (one with a probability of 2% and one with a probability of 10%).

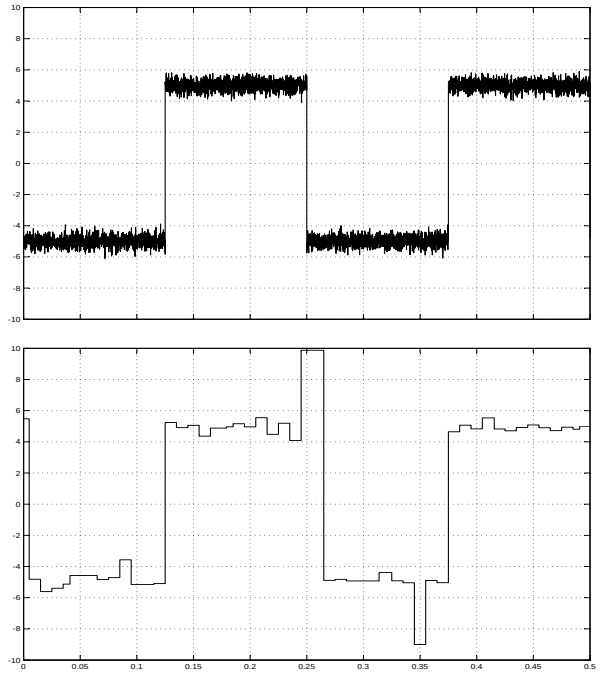

Figure 3. A noisy input signal and the corresponding node output with injected errors

4.1.2 Task and system model

Each node ran an identical replica of a single task with a period $P = 10ms$ and an execution requirement $C = 5ms$. This is because both the CMV and the VTV strategies expect parallel execution of the replica nodes. Except the timing errors, the only factor that may result in different output delivery times is the drift in the local clock from the global clock. The drift was simulated by allowing local periods slightly longer or shorter than $10ms$. The execution requirement was also scaled up or down based on the local period. Whenever the accumulated drift from the global clock, i.e. the accumulated sum of the difference between the local period and the real period, reaches the maximum admissible deviation from the real time ($\delta_{clock}/2 = 0.5ms$), the local clock is synchronized with the global clock. This is realized by running a synchronization period shorter or longer than the real period with a value equal to the accumulated difference. Figure 4 shows an example of a scenario with clock synchronization. The vertical arrows in the diagram indicate the beginning and the end of the periods (release times and the deadlines of the tasks). In this example, Node 3's clock runs in line with the real-time. However, Node 1's clock runs slower and Node 2's clock runs faster than the real-time. At $t = 39.5ms$, the drift of Node 2's clock reaches $\delta_{clock}/2 = 0.5ms$ and it executes a synchroniza-

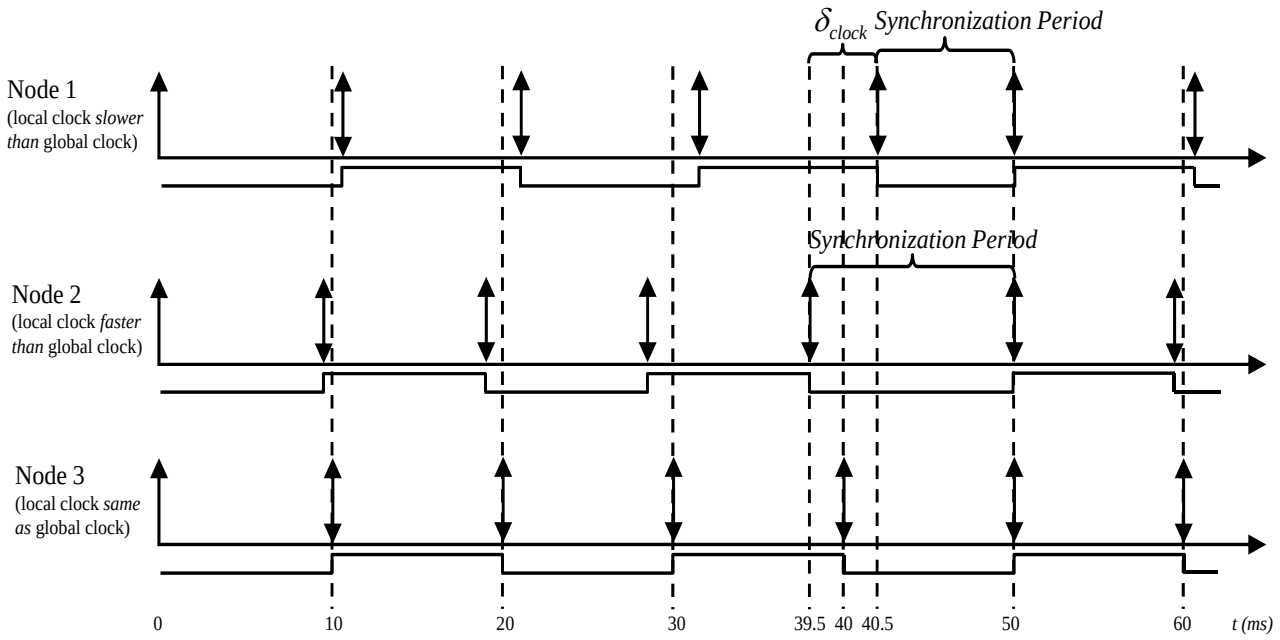

Figure 4. Clock synchronization in loosely synchronized systems

tion period $T_2^{synch} = 10.5ms$. Similarly, at $t = 40.5ms$, the drift of Node 1's clock reaches $\delta_{clock}/2 = 0.5ms$ and it executes a synchronization period $T_1^{synch} = 9.5ms$. At $t = 50ms$, all clocks are synchronized.

The nodes sample the input signals at the beginning of their periods and output the sampled values after the computation time scaled by the local period in case of error-free operation.

The time step used in the simulations was 100 simulation nanoseconds long, and each simulation was run for 100 simulation seconds. Increasing the simulation duration did not affect the statistical trend in the results, hence with the chosen error rates, a duration of 100 seconds for each run was shown to be adequate for the purpose of these experiments. The four error combinations (no error, only timing errors including omission errors, only value errors and both types of errors), the two error range combinations for both the time and the value errors, the two different error probabilities, the two different signals and the three different frequencies for each signal sum up to a total of 156 simulation runs. In addition, to show the effect of α on the performance of the evaluated voters, we separately ran the experiments for 5 different α values.

4.1.3 Voters

Both the voters that use the VTV strategy and the voter that uses the CMV strategy look for a majority in the value domain ($M_v = 3$). In addition, the voters that use the VTV strategy look for a majority in the time domain ($M_t = 3$). Furthermore, the voter that uses the VTV strategy and configured for Case 1 requires that the majority of replica outputs that match in the value domain are also timely at the same time. Two outputs are assumed to be matching in the value domain if the difference between them is less than or equal to the maximum admissible deviation in the value domain between any two replica outputs $\sigma = 0.2$ units, and they are assumed to be matching in the time domain if the difference in output delivery times is less than or equal to the maximum admissible deviation in time between any two nodes ($\delta_{clock} = 1ms$).

4.1.4 Perfect observer

The perfect observer module was used to calculate the *false positives* (FPR) and the *false negatives* (FNR) of each voting strategy. It compares the most recently received outputs from the voters with the output from the *reference node*. The outputs of the voters are assumed to be correct if the difference between them and the reference node is less than or equal to the maximum admissible deviation in the value domain from the ideal output $\sigma/2 = 0.1$ units. This task also uses the *error signals* from the voters, i.e. the signals that indicate the assumption violations, to determine the FPR and FNR.

f(Hz)	SIGNAL TYPE	CMV				VTV(Case 1)				VTV(Case 2)			
		FNR		FPR		FNR		FPR		FNR		FPR	
		(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)
1	Sine w.	0.09	0.39	1.17	3.92	0.01	0.03	4.84	25.51	0.06	0.11	2.72	16.41
	Square w.	0.05	0.57	1.04	3.64	0.02	0.09	5.06	25.87	0.04	0.21	2.80	16.23
4	Sine w.	0.56	5.36	1.08	3.07	0.02	0.31	4.53	21.56	0.31	1.19	2.92	14.74
	Square w.	0.17	2.22	1.38	3.72	0.03	0.19	5.01	25.38	0.08	0.65	3.11	16.96
16	Sine w.	1.74	11.27	0.8	2.49	0.09	0.9	3.10	14.89	0.93	3.81	2.38	11.83
	Square w.	0.73	6.47	1.24	3.18	0.05	0.34	4.55	20.98	0.44	1.87	3	14.35

Table 2. FPR and FNR for CMV and VTV with respect to the signal types and the signal frequencies

4.2 Results for $\alpha = 1$

Table 2 shows the FPR and the FNR of the evaluated voters for the given signals and the error probabilities assuming $\alpha = 1$. Regardless from the signal types, the signal frequencies and the error probabilities, the FPR of CMV is lower than that of VTV configured for Case 2, and the FPR of VTV configured for Case 2 is lower than that of VTV configured for Case 1. This is because the voters that use the VTV strategy identify more erroneous scenarios since, unlike the voter using the CMV strategy they can detect the assumption violations in the time domain, and some of those timing errors are not propagated into value errors. Among the voters using the VTV strategy, the FPR is higher for the one configured for Case 1 since, similarly, it signals a greater number of assumption violations, some of which are not propagated into value errors. On the other hand, it can be seen that the FNR of CMV is the highest among all the evaluated voting strategies and FNR of VTV configured for Case 1 is the lowest, which is much more critical than the difference in the FPR, as the FNR is the rate that indicates the errors that are neither masked nor signalled.

Table 3 shows the FPR and the FNR of the evaluated voters for the given error combinations and $\alpha = 1$. As expected, the FPR and FNR for all the voters are zero in the absence of errors. When only value errors are injected, the FPR and the FNR are identical for all voting strategies. This is because all strategies use the same criteria for detecting the value anomalies. However, when the injected errors are only in the form of timing errors (including the omission errors), the performance of the voters using the VTV strategy outperforms the voter using the CMV strategy by a great margin. This increase in the performance is still visible when the all types of errors are injected together.

Table 4 shows the FPR and the FNR of the evaluated voters for the given error magnitudes. Even when the injected errors become harder to detect due to their smaller magnitude, the VTV strategy outperforms the CMV strategy with respect to the FNR. However, we can see that the trend in the ratio of FNR's becomes slightly less distinct.

All of the above FPR and FNR values are derived by the perfect observer by comparing the voter outputs with the reference node output. As stated earlier, the voter output is identified as erroneous in the value domain if the difference is greater than $\sigma/2 = 0.1$ units. Figure 5 shows the ratio of CMV's FNR to VTV's FNR (configured for both Case 1 and Case 2) for the value errors that are greater than the values identified on the X-axis. This experiments show that the

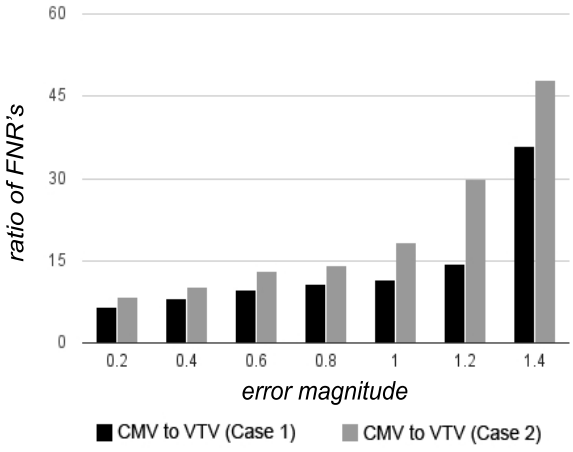

Figure 5. ratio of CMV's FNR to VTV's FNR (configured for Case 1 and Case 2) with increasing error magnitude

masking and signalling capability of CMV decreases relative to that of VTV as the error magnitude increases. Hence it provides a decreased level of fault-tolerance for the errors that are more critical. VTV performs better than CMV since the timeliness requirement of the signals limits the amount of deviation from the correct signal.

	CMV				VTV(Case 1)				VTV(Case 2)			
	FNR		FPR		FNR		FPR		FNR		FPR	
ERROR COMBINATION	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)
No errors	0	0	0	0	0	0	0	0	0	0	0	0
Only value errors	0.01	0.03	1.53	6.31	0.01	0.03	1.57	6.31	0.01	0.03	1.53	6.31
Only timing errors	0.29	1.48	0.17	0.29	0.02	0.13	3.12	15.29	0.16	0.53	1.49	7.9
Both value and timing errors	0.38	2.58	1.5	4.24	0.02	0.15	6.01	29.83	0.24	0.75	3.81	19.36

Table 3. FPR and FNR for CMV and VTV with respect to the error combinations

ERROR MAGNITUDE		CMV				VTV(Case 1)				VTV(Case 2)			
VALUE	TIMING	FNR		FPR		FNR		FPR		FNR		FPR	
ERRORS	ERRORS	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)
LARGE	LARGE	0.32	2.48	1.23	4.65	0.02	0.13	5.8	28.73	0.22	0.77	3.52	19.06
LARGE	SMALL	0.2	2.02	1.44	4.81	0.01	0.11	4.1	20.84	0.12	0.64	3.08	15.73
SMALL	LARGE	0.23	2.12	0.51	1.07	0.01	0.2	4.17	21.27	0.16	0.66	2.2	12.4
SMALL	SMALL	0.28	1.76	0.55	1.26	0.02	0.14	3.13	13.8	0.19	0.57	2.12	10.16

Table 4. FPR and FNR for CMV and VTV with respect to the error magnitude

4.3 Results for $\alpha \in [0.6, 1.4]$

Table 5 shows the FPR and the FNR of the evaluated voters for the given α values selected from the interval $[0.6, 1.4]$. Changing the value of α does not have any effect on the performance of CMV since no specific error detection is performed in the time domain. Nevertheless, the order among the FPR and FNR values for all the three types of voters are preserved for all the chosen values of α. For both configurations of VTV, as α increases, the FPR decreases and the FNR increases. This is because, for small values of α, a greater number of voter outputs are detected as potentially erroneous in the time domain. Among these detected outputs, there are scenarios both where these anomalies occur due to errors and due to clock drifts. If the anomalies are caused by errors, then their detection contributes to the reduction of FNR. If they are caused by clock drifts, then their detection causes an increase in the FPR.

5 Conclusions

In this paper, we have extended the VTV approach by enabling voter parameter tuning to achieve desired tradeoffs between the false positive and false negative rates. We have presented an evaluation of VTV, in comparison with the well-known voting strategy, CMV, both of which add the time dimension to the majority voting paradigm. The performed evaluation confirms that VTV outperforms CMV in all scenarios, showing a lower percentage of errors that are neither masked nor signalled. The experiments demonstrate the overcautious nature of VTV by showing a higher percentage of false positives due to the fact that it does not account the untimely values for arriving at a majority. The experiments also show that the ratio of the error masking and signalling capability of VTV to that of CMV increases as the magnitude of the value errors increase. Hence, VTV provides better error detection than CMV for value errors of greater magnitude which are originated by timing anomalies. This enhances the achieved dependability.

The goal of using redundancy schemes is to boost the reliability of the system to a level that the system specifications meet the dependability requirements. Our evaluations show that for a given redundancy scheme, it is possible to tune its performance by choosing (i) an appropriate δ (by adjusting the level of clock synchronization) and (ii) an appropriate α that in combination would enable reaching a desired level of FNR. On the other hand, the FPR needs to be bounded at a certain level so that the recovery actions taken due to the false positives would not jeopardize the real-time requirements. As a future work, we plan to develop a framework that would assist system designers to choose the degree of redundancy, together with the δ and α values that would both satisfy the dependability and real-time requirements at the same time. This will consist of a test environment for performing fault injection experiments and a sensitivity analysis tool to handle the false positives.

References

[1] A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. *Research Report N01145, LAAS-CNRS*, 2001.

	CMV				VTV(Case 1)				VTV(Case 2)			
	FNR		FPR		FNR		FPR		FNR		FPR	
α	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)	(2%)	(10%)
0.6	0.27	2.03	1.21	3.45	0.01	0.11	5.64	27.5	0.02	0.16	4.24	19.8
0.8	0.25	2.27	1.14	3.52	0.01	0.18	5.18	26.43	0.04	0.23	4	19.47
1	0.26	2.55	1.07	3.24	0.02	0.21	4.64	24.61	0.13	0.76	2.89	16.27
1.2	0.23	2.36	0.99	3.34	0.02	0.22	4.3	21.82	0.16	0.84	2.57	14.98
1.4	0.25	2.12	1.09	3.49	0.02	0.22	4.06	20.92	0.19	0.85	2.47	14.66

Table 5. FPR and FNR for CMV and VTV with respect to the α

[2] H. Aysan, S. Punnekkat, and R. Dobrin. VTV - a voting strategy for real-time systems. *IEEE Pacific Rim International Symposium on Dependable Computing*, pages 56–63, 2008.

[3] D. Blough and G. Sullivan. A comparison of voting strategies for fault-tolerant distributed systems. *Proceedings of the 9th Symposium on Reliable Distributed Systems*, pages 136–145, 1990.

[4] A. Bondavalli and L. Simoncini. Failure classification with respect to detection. *Proceedings of the 2nd IEEE Workshop on Future Trends in Distributed Computin*, pages 47–53, 1990.

[5] S. S. Brilliant, J. C. Knight, and N. G. Leveson. The consistent comparison problem in N-version software. *IEEE Transactions on Software Engineering*, 15(11):1481–1484, 1989.

[6] L. Chen and A. Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. *Proceedings of the 25th International Symposium on Fault-Tolerant Computing, ' Highlights from Twenty-Five Years'.*, pages 113–119, 1995.

[7] P. Ezhilchelvan, J.-M. Helary, and M. Raynal. Building responsive TMR-based servers in presence of timing constraints. *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 267–274, 2005.

[8] V. D. Florio, G. Deconinck, and R. Lauwereins. The EFTOS voting farm: A software tool for fault masking in message passing parallel environments. *Proceedings of the 24th Euromicro Conference*, 1:379–386, 1998.

[9] F. D. Giandomenico and L. Strigini. Adjudicators for diverse-redundant components. *Proceedings of the 9th Symposium on Reliable Distributed Systems*, pages 114–123, 1990.

[10] A. Grnarov, J. Arlat, and A. Avizienis. Modeling of software fault-tolerance strategies. *Proceedings of the 11th Annual Pittsburgh Modeling and Simulation Conference*, pages 571–578, 1980.

[11] H. Kopetz. Fault containment and error detection in the time-triggered architecture. *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems*, pages 139–146, 2003.

[12] J. Lala, L. Alger, S. Friend, G. Greeley, S. Sacco, and S. Adams. An analysis of redundancy management algorithms for asynchronous fault tolerant control systems. *Research Report NASA-TM-100007, NASA*, 1987.

[13] J. Lala and R. Harper. Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, 82(1):25–40, 1994.

[14] J.-C. Laprie. Dependable computing and fault-tolerance: Concepts and terminology. *Proceedings of the 25th International Symposium on Fault-Tolerant Computing, ' Highlights from Twenty-Five Years'.*, 1995.

[15] G. Latif-Shabgahi, J. Bass, and S. Bennett. A taxonomy for software voting algorithms used in safety-critical systems. *IEEE Transactions on Reliability*, pages 319–328, 2004.

[16] P. Lorczak, A. Caglayan, and D. Eckhardt. A theoretical investigation of generalized voters for redundant systems. *Proceedings of the 19th International Symposium on Fault-Tolerant Computing, FTCS-19. Digest of Papers.*, pages 444–451, 1989.

[17] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *Journal of Research and Development*, 6:200–209, 1962.

[18] J. V. Neuman. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, pages 43–98, 1956.

[19] D. Powell. Failure mode assumptions and assumption coverage. *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*, pages 386–395, 1992.

[20] D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabejac, and A. Wellings. GUARDS: a generic upgradable architecture for real-time dependable systems. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):580–599, 1999.

[21] K. Ravindran, K. Kwiat, and A. Sabbir. Adapting distributed voting algorithms for secure real-time embedded systems. *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, pages 347–353, 2004.

[22] K. Ravindran, K. Kwiat, A. Sabbir, and B. Cao. Replica voting: a distributed middleware service for real-time dependable systems. *Proceedings of the 1st International Conference on Communication System Software and Middleware*, pages 1–7, 2006.

[23] J. Rushby. Formal methods and the certification of critical systems. *Computer Science Laboratory, SRI International, Tech. Rep CSL-93-7*, 1993.

[24] K. Shin and J. Dolter. Alternative majority-voting methods for real-time computing systems. *IEEE Transactions on Reliability*, 38(1):58–64, 1989.