Improving Reliability of Real-Time Systems through Value and Time Voting

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

Abstract

Critical systems often use N-modular redundancy to tolerate faults in subsystems. Traditional approaches to N-modular redundancy in distributed, loosely-synchronised, real-time systems handle time and value errors separately: a voter detects value errors, while watchdog-based health monitoring detects timing errors. In prior work, we proposed the integrated Voting on Time and Value (VTV) strategy, which allows both timing and value errors to be detected simultaneously. In this paper, we show how VTV can be harnessed as part of an overall fault tolerance strategy and evaluate its performance using a well-known control application, the Inverted Pendulum. Through extensive simulations, we compare the performance of Inverted Pendulum systems which employs VTV and alternative voting strategies to demonstrate that VTV better tolerates well-recognised faults in this realistically complex control problem.

1 Introduction

Dependability is important in safety-critical systems, where a failure (systematic or random) could cause harm to humans or the environment. Since it is difficult to demonstrate that all faults have been completely removed, achieving adequate dependability frequently requires tolerating some faults. In the common *N-modular redundancy* fault tolerance mechanism, critical functions are replicated [22]. A *voter* compares outputs from all replicated channels to determine both whether a fault has occurred and what result the system should use.

Many safety-critical systems are real-time systems: if the system misses too many deadlines, it will enter a hazardous state and an accident might occur [4]. Distributed real time systems might be either *tightly synchronized* or *loosely synchronized*. Loose synchronization is attractive and often used because it reduces overheads, complexity, and reliance on the synchronization mechanism itself [8]. However, most existing N-modular redundant voting strategies assume tight synchronization and focus solely on tolerating anomalies in the value domain [12]. In these systems, other mechanisms, such as timing watchdogs, are used to detect timing errors. These strategies wait until all signals arrive, or watchdog timers fire for all late or omitted signals, before producing output. However, a voter can in fact provide output as soon as it has received sufficient inputs. Such an approach can mask timing faults that would otherwise harm system dynamics. This paper assumes loose synchronisation due to its benefits, e.g. its relative simplicity compared to tighter synchronisation, and the fact that it prevents the global time base becoming too critical.

In previous work, we proposed an integrated voting strategy based on both value and time [2]. We call this approach Voting on Time and Value (VTV). Traditional approaches to N-modular redundancy capture timing errors only when a deadline is exhausted. In contrast, VTV detects timing *deviations* between the replica outputs as they occur, yielding more reliable and earlier detection of faults in loosely synchronized systems. Replicated channels might produce outputs within their deadlines at quite different times (e.g., one soon after their release and one near the deadline) because of a problem that could cause failures later. VTV can detect such situations earlier than traditional approaches. This facilitates quicker recovery, such as resetting the offending channel, and increases the likelihood that the deadlines are met. Our overarching claim is that systems using VTV rather than traditional approaches can be more capable of delivering correct service despite credible failures.

In prior work, we evaluated VTV using abstract sim-

ulation [2]. In this paper, we report a more thorough, more realistic evaluation of VTV's ability to handle transient faults. We have assessed VTV's performance through extensive simulation of a well-known real-time control system. The inverted pendulum is a challenging control problem where subtle undetected failures lead to undesirable consequences [3] and thus representative of classic safety-critical systems. The evaluation framework has two principal benefits. Firstly, the same scenarios can be run across a number of state of the art approaches to give a comparative evaluation. Secondly, the scenarios can be controlled, thus allowing the approaches to be compared as key parameters (e.g. type, likelihood and size of errors) are adjusted such that we understand the relative performance of the algorithms. In this paper, we build on our prior work with four new contributions:

1. We compare fault detection approaches in terms of their system-level effect on the inverted pendulum (e.g., accuracy and responsiveness)
2. We assess VTV's ability to mask and detect transient faults in comparison to that of voting on value with or without a watchdog timer
3. We demonstrate how this enables the system to be more dependable in the presence of credible failures
4. We discuss how VTV's strengths can be harnessed (and its weaknesses overcome) to create a complete approach to tolerating transient faults and managing permanent faults

In Section 2, we describe the dependability model we assume and define the redundancy approach in which VTV could be used. We discuss related work in Section 3 and define VTV and the alternative voting strategies in Section 4. We present evaluation results in Section 5 and conclude the paper in Section 6.

2 Dependability Model and Redundancy Approach

We follow the dependability concepts originally introduced in [1, 6, 17]. Faults in replicated channels might lead to errors in the visible state of the subsystems that use these channel outputs. Replication is meant to mask some of these errors, preventing them from causing system failure. Errors in individual channel outputs might be either *time* errors or *value* errors and might also be either *transient* or *permanent*.

2.1 Time and Value Errors

We model our system using the following variables:

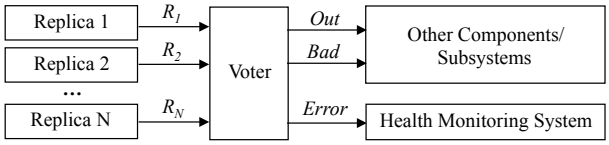

Figure 1. Voter inputs and outputs

R_i	The output of the i^{th} replicated channel, with $R_i =< v_i, t_i >$
v^*	The value that an ideal channel and voter would produce
σ	The maximum that the voter's output may deviate from v^*
t^*	The design target time at which the channel should produce output
δ	The design target for maximum jitter in the channel output

These definitions simplify our earlier model [2]: we have standardized on a one-sided definition of tolerances δ and σ and eliminated a variable that is not used in this paper. Designers derive σ from system safety requirements and select t^* and δ given the timing requirements on the voter's output and the computation time of the voter itself. Given these parameters, we define replica R_i's output as correct if and only if it is both within σ of v_i^* and delivered within δ of t_i^*:

$$(v_i^* - \sigma \leq v_i \leq v_i^* + \sigma) \wedge (t_i^* - \delta \leq t_i \leq t_i^* + \delta)$$

2.2 Transient and Permanent Errors

The erroneous visible state of a replicated component is a fault at the system level. These errors might be either transient or permanent. We consider a replicated channel's erroneous state to be transient if and only if the channel would clear the error without system-level intervention. If restoring error-free operation requires resetting the channel's hardware or software, or repairing or replacing a line replaceable unit, we consider the error permanent.

2.3 The Voter Component and Its Design Goals

Figure 1 illustrates the system that we assume. Each replicated channel is implemented on a separate processor. Channels begin processing when they receive input and send output to the voter when they finish. Unlike tightly-synchronized systems, we allow individual clocks to drift to a degree that can be achieved using relatively inexpensive software-based clock synchronization.

The primary purpose of the voter is to mask errors in the redundant channels. However, it is also a part of the system's mechanism for remediating permanent errors in the channels. If the voter finds a consensus, *Out* gives it and the voter signals $\neg Bad$ to indicate that *Out* can be trusted. Otherwise, *Out* gives a best guess and the voter signals *Bad*. To facilitate health monitoring and management, the voter reports any *Errors* it detects in the replicated channels' output.

The voter's requirements vary from system to system. For example, fail operational systems depend on the voter producing acceptable *Out* values even when there is no consensus. Broadly speaking, the voter is subject to three design goals:

G1. Produce good *Out* values as frequently as possible.

G2. Minimize false negative $\neg Bad$ and false positive *Bad* signals.

G3. Report as many detectable *Errors*, as early as possible.

Health Monitoring and Management. The health monitoring and management strategy assumed has two design goals:

1. *Permanent error detection.* In this paper, the error monitoring and management strategy identifies a channel to be affected by a permanent error if the voter reports p consecutive errors from that channel.
2. *Recovery from permanent errors.* In this paper, we perform recovery by resetting the erroneous channel. Our simulations model a reset period of r voting periods during which the channel produces no output.

3 Related Work

Many different voting strategies have been proposed to tolerate faults and researchers have been addressing different aspects of these strategies depending on the characteristics of the target system.

Voter Dependability. One disadvantage of N-modular redundancy is that the approach introduces a new single point of failure, namely the voter [13]. One approach to achieving a highly reliable voter is to construct it of very simple electronic circuits. This approach is useful in *exact* voting, i.e. where all non-faulty replicates produce bitwise-identical output, particularly when output is generated at high frequencies [13]. However, in some systems non-faulty replicates might produce slightly different values [7]. For example, replicates might use data from different sensors or a different calculation algorithm. It is more difficult to produce simple voting hardware for these more complex *inexact* voting applications [13]. Another approach to achieving a highly reliable voter is to replicate the voter [13, 9, 15]. In the assessment of VTV we report in Section 5, a single voter implemented in software. The complexity of VTV might make a hardware implementation complicated. However, we are unaware of any reason why the VTV approach could not be implemented using a replicated voter.

Definitions of Consensus. Researchers have proposed different definitions of consensus and strategies for deriving an output from that consensus [5, 10]. *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 [11, 14]. *Median* voters output the median value [13]. *Average* voters output the mean. Some practitioners have advocated the use of geometric means rather than arithmetic means in some circumstances, arguing that the former is less sensitive to outliers [21]. The implementation of VTV that we used in this paper reports the mean of a simple-majority consensus. However, VTV approach can very well be implemented using a different definition of consensus or strategy for deriving output from it.

Other Techniques That Address Timing Errors. Researchers have proposed majority voting techniques that are able to implicitly handle the errors in the time domain [18, 19, 20]. In the Quorum Majority Voting (QMV) and Compare Majority Voting (CMV) techniques, the voter builds a consensus as soon as it receives a quorum or a majority of responses. These approaches aim to mask late timing errors and both provide outputs within a bounded time interval. Both also assume that the number of timing errors within a certain period does not exceed an allowed threshold. As long as this assumption holds, QMV and CMV can detect value errors as well as existing approaches. However, neither QMV nor CMV can detect violations of this assumption. In contrast, VTV explicitly detects these violations and reports them for health monitoring purposes. Because QMV and CMV do not detect timing assumption violations, they must be complemented by health monitoring for the same. Moreover, when their timing assumptions are violated, they might produce incorrect values.

4 Voting Strategies

We have evaluated VTV by comparing it to two alternatives: voting on value (V) and voting on value

with a watchdog timer (VWDT). These approaches represent the typical approaches reported in the literature [13] and those approaches used in industry. In this section, we describe all three approaches. We have chosen these two alternatives to investigate how an event-triggered approach (VTV) compares to time-triggered approaches with and without timing error detection capabilities (V and VWDT, respectively).

In the following descriptions, we assume the parameters used in our evaluation. For example, while VTV is applicable to any number of replicated channels $N \geq 3$, we describe VTV as implemented for triple modular redundancy (i.e., $N = 3$). While it should be possible to implement different strategies for deriving *Out* from a consensus, we describe VTV as implemented using an arithmetic mean strategy.

Voting on Time and Value (VTV). We introduced VTV in prior work [2] (Algorithm 1). This approach is event triggered: the voter considers each R_i as it arrives and produces output as soon as there is a consensus among the available inputs. Table 1 defines 'consensus' and the output produced for each input scenario. Because VTV is event-triggered, if it receives no input it will produce no output. If it is necessary to produce output even in these cases (e.g., because an actuator would drift otherwise) the output component can be made to latch the voter's last output.

Algorithm 1: Voting on Value and Time (VTV)

```
Inputs: sensor outputs with time stamps
Outputs: Out, Bad, Error
while true do
    if consensus among available and timely outputs then
        Out = consensus;
        Bad = false;
    else
        if there are three timely outputs then
            Out = last good value;
        else if there are two timely outputs then
            Out = randomly choose one;
        else if there is one timely output and all three channels are active (the other two channels can form a majority) then
            Out = none;
        else if there is one timely output then
            Out = the timely output;
        Bad = true;
    end
    if detected error then
        Error = detected error;
    end
end
```

Voting on Value (V). Von Neumann introduced the basic Voting on Value majority voting strategy, which can mask up to n value errors in $3n$ inputs [24]. We implement this strategy using a fixed-period, time-triggered approach. Algorithm 2 shows the details of this voting procedure.

Algorithm 2: Voting on Value (V)

```
Inputs: latest sensor outputs
Outputs: Out, Bad, Error
if consensus among available outputs (active channels) then
    Out = consensus;
    Bad = false;
else
    Out = last good value;
    Bad = true;
end
if detected error then
    Error = detected error;
end
```

Voting on Value with Watchdog Timers (VWDT). Voting on Value with Watchdog Timers extends the V strategy by using watchdog timers to mask and detect timing anomalies. Untimely inputs are excluded from each voting round. We implement this strategy as shown in Algorithm 3.

Algorithm 3: Voting on Value with Watchdog Timers (VWDT)

```
Inputs: latest sensor outputs with time stamps
Outputs: Out, Bad, Error
if consensus among available and timely outputs then
    Out = consensus;
    Bad = false;
else
    if there are two timely outputs then
        Out = randomly choose one;
    else if there is one timely output then
        Out = the timely output;
    else
        Out = last good value;
    end
    Bad = true;
end
if detected error then
    Error = detected error;
end
```

Table 1. The Voting on Time and Value (VTV), Voting on Value (V), and Voting on Value with Watchdog Timers (VWDT) voting approaches

Voter	Number of Active Channels	Input Scenario	*Out*	*Bad*	*Error*
VTV	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge M_{b,c} \wedge M_{a,c}$	mean(v_a, v_b, v_c)	false	none
	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge M_{b,c} \wedge \neg M_{a,c}$	median(v_a, v_b, v_c)	false	none
	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	mean(v_a, v_b)	false	c value
	3	$T_a \wedge T_b \wedge T_c \wedge \neg M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	last good value	true	none
	3	$T_a \wedge T_b \wedge \neg T_c \wedge M_{a,b}$	mean(v_a, v_b)	false	c time
	3	$T_a \wedge T_b \wedge \neg T_c \wedge \neg M_{a,b}$	randomly v_a or v_b	true	c time
	3	$T_a \wedge \neg T_b \wedge \neg T_c$	none	false	a time
	3	no input	none	false	none
	2	$T_a \wedge T_b \wedge M_{a,b}$	mean(v_a, v_b)	false	none
	2	$T_a \wedge T_b \wedge \neg M_{a,b}$	randomly v_a or v_b	true	none
	2	$T_a \wedge \neg T_b$	v_a	true	none
	2	no input	none	false	none
	1	T_a	v_a	false	none
	1	no input	none	false	none
V	3	$M_{a,b} \wedge M_{b,c} \wedge M_{a,c}$	mean(v_a, v_b, v_c)	false	none
	3	$M_{a,b} \wedge M_{b,c} \wedge \neg M_{a,c}$	median(v_a, v_b, v_c)	false	none
	3	$M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	mean(v_a, v_b)	false	c value
	3	$\neg M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	last good value	true	none
	2	$M_{a,b}$	mean(v_a, v_b)	false	none
	2	$\neg M_{a,b}$	last good value	true	none
	1	-	v_a	false	none
VWDT	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge M_{b,c} \wedge M_{a,c}$	mean(v_a, v_b, v_c)	false	none
	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge M_{b,c} \wedge \neg M_{a,c}$	median(v_a, v_b, v_c)	false	none
	3	$T_a \wedge T_b \wedge T_c \wedge M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	mean(v_a, v_b)	false	c value
	3	$T_a \wedge T_b \wedge T_c \wedge \neg M_{a,b} \wedge \neg M_{b,c} \wedge \neg M_{a,c}$	last good value	true	none
	3	$T_a \wedge T_b \wedge \neg T_c \wedge M_{a,b}$	mean(v_a, v_b)	false	c time
	3	$T_a \wedge T_b \wedge \neg T_c \wedge \neg M_{a,b}$	randomly v_a or v_b	true	c time
	3	$T_a \wedge \neg T_b \wedge \neg T_c$	v_a	true	b, c time
	3	no input	last good value	true	a, b, c time
	2	$T_a \wedge T_b \wedge M_{a,b}$	mean(v_a, v_b)	false	none
	2	$T_a \wedge T_b \wedge \neg M_{a,b}$	randomly v_a or v_b	true	none
	2	$T_a \wedge \neg T_b$	v_a	true	b time
	2	no input	last good value	true	a,b time
	1	T_a	v_a	false	none
	1	no input	last good value	true	a time

Key. In each round of voting, we refer to one of the channels as a, another as b, the last as c, and the time of the first arrival as t_0. For VTV, we define each channel as timely (T_i) if and only if it arrives within 2δ of t_0: $T_i \equiv t_i \leq t_0 + 2\delta$. For VWDT, we define each channel as timely (T_i) if and only if R_i arrives before the timer fires: $T_i \equiv t_i \leq t^* + \delta$. For all voters, we define two channels as matching if and only if their values differ by less than 2σ: $M_{i,j} \equiv \text{abs}(v_i - v_j) \leq 2\sigma$.

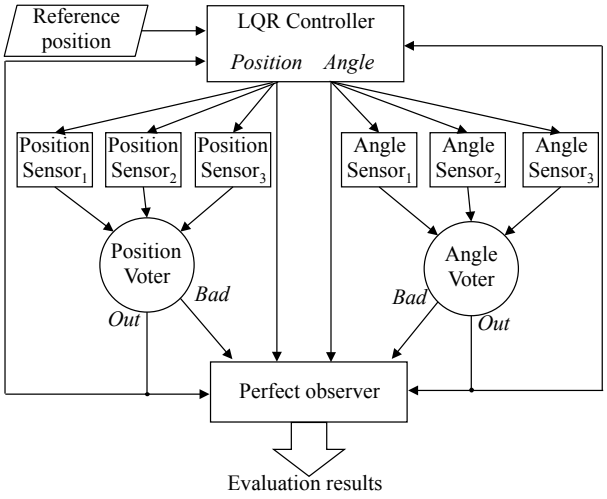

Figure 2. Simulation setup

5 Evaluation

We evaluate the three approaches using an inverted pendulum simulated in Matlab/Simulink. In this section, we assess how well VTV meets the first two of three goals identified in Section 2.3 compared to V and VWDT. We discuss the third goal in Section 6 but leave a more detailed evaluation for future work.

5.1 Study Design

Figure 2 illustrates the setup. The pendulum is assumed to be mounted on a cart that moves on a linear track. The control objective is to move the cart to a reference position while keeping a pendulum, attached to its top by a pivot, upright. Triplicated position sensors monitor the cart's position along the track. Triplicated angle sensors monitor the angle between the pendulum and the cart's top. Voting mask transient errors in both types of sensor. A linear-quadratic regulator (LQR) feedback controller produces control output to the cart's motors.

The sensors sample position and angle periodically. Position sensors nominally report a new position at intervals of 20 millseconds. Angle sensors nominally report a new position at intervals of 10 milliseconds. We inject random value and timing errors (delays) into their outputs to simulate the faults that real sensors experience. Value and timing errors might have either $(s)mall$ or $(l)arge$ magnitude and $(l)ow$ or $(h)igh$ probability or $(n)ot\ at\ all$. To explore the effect of these variables, we simulated each permutation of

$$Voter \times ValueMagnitude \times TimeMagnitude \times Prob$$

where $Voter$ is the set $\{VTV, V, VWDT\}$, $ValueMagnitude$ and $TimeMagnitude$ are the set $\{s, l\}$, and $Prob$ is the set of value and time error probability combinations $\{(l,l),(l,n),(n,l),(h,h),(h,n),(n,h)\}$.

Value Errors. At each period, each sensor i collects a sample v_i that it will report unless its output is omitted. This sample is computed from the simulator's ground truth v^* as $v_i = v^* + a_v b_v$, where a_v controls whether an error occurs and b_v gives its magnitude. For position sensors, the magnitude is an integer length unit where the track runs from -9 to 9. For angle sensors, the magnitude is an integer angle in degrees. We randomly select $a_v = 1$ with probability 0% (not at all), 1% (low), or 10% (high) and 0 otherwise. We randomly select b_v from the range $[-1..1]$ (small) or $[-5..5]$ (large) with uniform probability.

Timing Errors. Each sensor i nominally reports its sample at time t_i unless its output is *delayed* or *omitted*. We compute the sample time from the nominal sample time t_s as $t_i = t_s + a_t b_t + j$, where a_t controls whether an error occurs, b_t gives the magnitude of the delay in milliseconds, and j gives the jitter in milliseconds. We randomly select $a_t = 1$ with probability 0% (not at all), 1% (low), or 10% (high) and 0 otherwise. We randomly select b_t from the range [1..100] (small) or [1..1000] (large) with uniform probability. We randomly select j from $\{-1, 0, 1\}$ with uniform probability. As each sensor goes to report its sample, we randomly decide whether to omit the output with probability 0% (not at all), 0.1% (low), or 1% (high). If an output is omitted, the next period's output is also omitted (i.e., we are simulating a two-period sensor outage).

System Parameters. We use value tolerance $\sigma = 0.02$ length units for position sensors and $\sigma = 0.02$ degrees for angle sensors. We use time tolerance $\delta = 1$ milliseconds for both sensor types. Our V and VWDT voters run $\delta = 1$ millisecond after the nominal time that the sensors report their samples in each sensor period. A channel is assumed to be permanently failed if the voter reports $p = 3$ consecutive errors from that channel. Health monitoring and management strategy resets a permanently failed channel to attempt recovery which is modeled by not producing output for $r = 100$ voting periods. VTV is event triggered by design. Voters feed their consensus value to the controller as soon as consensus is formed. Due to its event triggered nature, comparing VTV with V and VWDT is not a trivial task. Our main goal in this study is to show the benefits and drawbacks of each approach by covering a wide range of different scenarios. We simulated each

Table 2. Pendulum MTBF in seconds. Results show data from a set of simulation runs that include both time and value errors injected with high magnitude and probability. We provide data from simulations with one sensor (no voting) for comparison.

VTV	V	VWDT	Single sensor
9033.7	2940.8	517.4	4.3

Table 3. Statistical analysis of VTV's MTBF performance versus that of V, VWDT, and a single sensor with no voting. We computed p-values using the Mann-Whitney U test [16]. We computed a-values using the Vargha-Delaney a-test [23]. MTBF ratios are given as the ratio of VTV's MTBF to that of the voter in comparison.

VTV Versus	p-value	a-value	MTBF Ratio
V	$1.32 \cdot 10^{-04}$	0.79	3.07
VWDT	$1.21 \cdot 10^{-10}$	0.98	17.46
single sensor	$3.02 \cdot 10^{-11}$	1.00	2108.14

permutation described above for 100 seconds, having first checked that longer runs produced similar results.

Perfect Observer and Evaluation Metrics. We use a perfect observer module to evaluate the voters' *Out* and *Bad* outputs by comparing these to the ground truth of the simulation. An *Out* signal is counted correct if and only if it differs from ground truth by no more than σ. A *Bad* signal is counted as a *true positive* if $Bad \wedge \text{correct}(Out)$, a *false positive* if $Bad \wedge \neg\text{correct}(Out)$, a *true negative* if $\neg Bad \wedge \neg\text{correct}(Out)$, or a *false negative* if $\neg Bad \wedge \text{correct}(Out)$.

5.2 Results

System-Level Performance. The simulated inverted pendulum remains upright longer using VTV than when using either V or VWDT. Table 2 compares the mean time between failures (i.e., pendulum falls) (MTBF) across approaches. Table 3 presents a statistical analysis showing that VTV's effect on MTBF is significant.

Producing Good *Out* Values. VTV generally satisfies design goal G1 better than either V or VWDT.

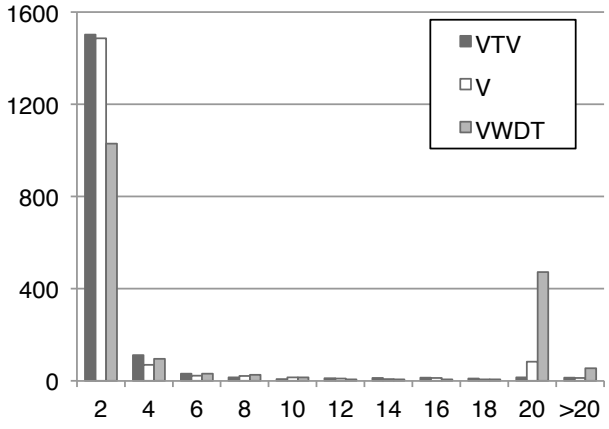

Figure 3. Distribution of lengths of periods for which *Out* was incorrect in the value domain (in milliseconds)

Table 4 and Table 5 show that VTV's *Out* is more frequently correct than that of V or VWDT, but the difference is very subtle. However, Figure 3 also shows that the majority of the errors produced by VTV are much shorter than those produced by others. As V and VWDT works in a time triggered way, they may not be able to act immediately upon error occurrences and may need to wait until the next scheduled voting round. This is the reason of the peak in the distribution for the errors whose lengths are equal to the period of the voting task. This means that when VTV outputs an incorrect *Out*, it corrects itself faster than V or VWDT. Figure 4 shows the distribution of magnitude of value deviations in *Out* from the ideal value and indicates that VTV's *Out* values are generally closer to ground truth than those produced by V or VWDT. Readers should note that only the last columns for each respective voter show the outputs that are considered as incorrect in the value domain ($|v_i^* - v_i| > 0.02$). Figure 5 shows the distribution of incorrect values in *Out* for different maximum jitter, caused by different levels of clock synchronisation, and maximum allowed drift (δ) scenarios. The figure shows that VTV performs better than V and VWDT, outputting fewer incorrect outputs, as the clock synchronisation gets looser.

Minimising False Negative and False Positive *Bad* Signals. VTV generally satisfies design goal G2 better than either V or VWDT. Table 6 shows that VTV's *Bad* output contains consistently fewer false positives and a consistently lower false positive rate than V or VWDT. However, Table 7 shows VTV's

Table 4. Proportion of simulation time during which each voter's *Out* was correct. Figures (in %) combine data from simulations of all *Probability* combinations.

Magnitude		Position Sensor			Angle Sensor		
Value	Time	VTV	V	VWDT	VTV	V	VWDT
small	small	97.82	97.88	97.51	96.52	96.36	96.12
large	small	97.67	97.71	98.01	96.84	96.72	96.77
small	large	97.79	97.98	97.34	96.90	96.60	96.45
large	large	98.08	98.03	97.80	97.12	96.68	96.52

Table 5. Proportion of simulation time during which each voter's *Out* was correct. Figures (in %) combine data from small and large magnitude simulations.

Errors Injected			Position Sensor			Angle Sensor		
Value	Time	Probability	VTV	V	VWDT	VTV	V	VWDT
✓		low	99.35	99.45	99.48	98.95	99.08	99.12
	✓	low	99.37	99.46	99.47	99.02	99.03	99.09
✓	✓	low	99.34	99.40	99.47	99.08	98.97	99.10
✓		high	98.70	98.79	98.76	98.11	98.05	98.04
	✓	high	98.24	98.06	98.73	97.29	96.68	98.11
✓	✓	high	96.34	96.40	95.87	94.61	94.21	93.82

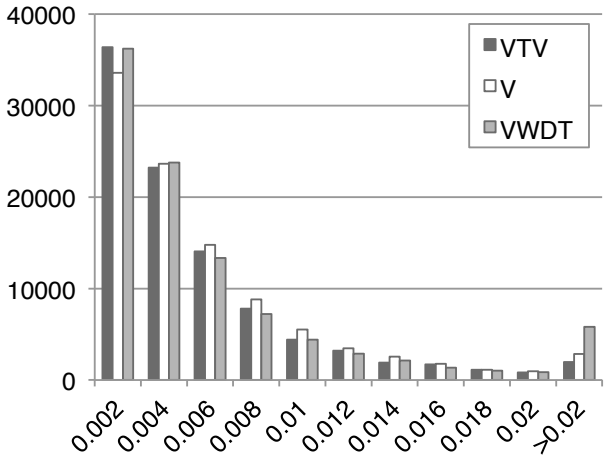

Figure 4. Distribution of magnitudes of value deviation in *Out* from the ideal value (in degrees)

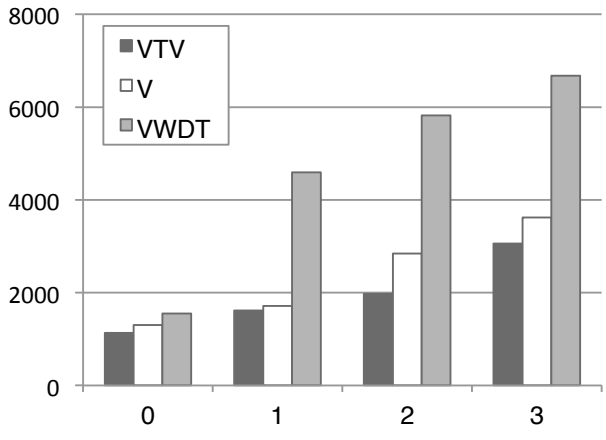

Figure 5. Distribution of incorrect *Out* with respect to maximum allowed clock drift (in milliseconds)

higher false negative rate. This is partly due to VTV's better *Out* values and partly due to VTV missing slightly more of its incorrect *Out* values (i.e., FN + FP is lower for VTV in almost every case).

Our simulations show that, for the simulated system, VTV outperforms V and VWDT in terms of keeping the pendulum upright longer. The simulations show mixed results for the simulated voters in satisfying the voter design goals G1 and G2 defined in Section 2.3 with a slight advantage for VTV.

Table 6. *Bad* **false positives with respect to error magnitude. FP and TN figures are given as percentages of total voter outputs. FP ratio given in percent. Figures combine data from simulations of all** *Probability* **combinations.**

Mag.		VTV		V		VWDT		FP Ratio		
V/T	Sensor	FP	TN	FP	TN	FP	TN	VTV	V	VWDT
s/s	angle	0.9	98.1	1.9	97.5	4.6	94.4	0.9	1.9	4.6
s/l	angle	0.9	98.2	1.7	97.6	4.4	94.7	0.9	1.7	4.4
l/s	angle	1.0	98.1	1.9	97.6	4.2	94.9	1.0	1.9	4.3
l/l	angle	1.1	98.1	2.0	97.4	4.7	94.3	1.1	2.0	4.8
s/s	position	1.0	98.4	1.4	98.2	4.0	95.2	1.0	1.4	4.0
s/l	position	1.2	98.2	1.6	98.0	4.4	94.8	1.2	1.6	4.5
l/s	position	1.0	98.2	1.7	97.9	4.1	95.4	1.0	1.7	4.1
l/l	position	1.0	98.6	2.2	97.5	4.2	95.2	1.0	2.2	4.2

Table 7. *Bad* **false negatives with respect to error magnitude. TP and FN figures are given as percentages of total voter outputs. FN ratio given in percent. Figures combine data from simulations of all** *Probability* **combinations.**

Mag.		VTV		V		VWDT		FN Ratio		
V/T	Sensor	TP	FN	TP	FN	TP	FN	VTV	V	VWDT
s/s	angle	0.27	0.81	0.42	0.23	0.88	0.13	75.3	35.7	12.9
s/l	angle	0.17	0.75	0.49	0.21	0.82	0.13	81.4	30.0	13.8
l/s	angle	0.19	0.75	0.40	0.14	0.76	0.10	80.1	26.2	11.2
l/l	angle	0.17	0.67	0.45	0.11	0.86	0.09	80.1	19.6	9.5
s/s	position	0.10	0.52	0.26	0.10	0.66	0.12	83.9	27.8	15.4
s/l	position	0.08	0.53	0.31	0.11	0.69	0.08	86.9	26.2	10.4
l/s	position	0.16	0.59	0.35	0.11	0.47	0.05	78.7	23.9	9.6
l/l	position	0.15	0.28	0.27	0.06	0.66	0.03	65.1	18.2	4.3

6 Conclusions

In this paper, we report on a simulation-based evaluation VTV's ability to produce correct output in the face of errors and to accurately report bad outputs. Our results show that, most of the time, VTV outperforms periodic voting in the value domain with and without watchdog timers. VTV restores service more quickly after errors and falsely signals bad output less frequently than either V or VWDT. Using VTV, our simulated inverted pendulum remains upright longer despite the injected value and time errors. Even though the total durations during which the voters' outputs were incorrect were quite similar, VTV's errors were more in number than those of V and VWDT but they were much shorter. We believe that this is the main advantage of using VTV and the fundamental reason why the pendulum remains upright longer.

We present MTBF data for only one configuration of errors because time constraints precluded simulating each permutation for a sufficient duration. To overcome these constraints, we attempted to run the simulations with higher error likelihood. When we injected only value errors, all voters produced similar results due to similarity of the voting approaches in the value domain alone. When we injected only timing errors with a higher error rate, the effect of repeated health-management resets seemed to dominate the effects of the voters' accuracy. We leave precise assessment of MTBF under these other conditions for for future work.

Because it is event triggered, VTV can pass data on to consumers faster than periodic designs. However, for the same reason, VTV has more jitter in its outputs. If there are multiple timing errors in its inputs, it might even deliver multiple outputs or no output in a given period. Our assessment suggests that its advantages outweigh these drawbacks.

References

[1] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, January–March 2004.

[2] H. Aysan, S. Punnekkat, and R. Dobrin. VTV — A voting strategy for real-time systems. In *Proc. of the IEEE Pacific Rim Intl. Symposium on Dependable Computing*, pages 56–63, 2008.

[3] S. Bak, D. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha. The system-level simplex architecture for improved real-time embedded system safety. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 99–107, 2009.

[4] I. Bate and A. Burns. An integrated approach to scheduling in safety-critical embedded control systems. *Real-Time Systems Journal*, 25(1):5–37, 2003.

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

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

[7] 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, November 1989.

[8] G. J. Davis. An analysis of redundancy management algorithms for asynchronous fault tolerant control systems. Technical Memorandum NASA-TM-100007, National Aeronautics and Space Administration, September 1987.

[9] V. De Florio, G. Deconinck, and R. Lauwereins. The EFTOS voting farm: A software tool for fault masking in message passing parallel environments. In *Proc. of the 24th Euromicro Conference*, pages 379–386, August 1998.

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

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

[12] H. Kopetz. Fault containment and error detection in the time-triggered architecture. In *Proc. of the 6th Intl. Symposium on Autonomous Decentralized Systems*, pages 139–146, April 2003.

[13] G. Latif-Shabgahi, J. M. Bass, and S. Bennett. A taxonomy for software voting algorithms used in safety-critical systems. *IEEE Transactions on Reliability*, 53(3):319–328, September 2004.

[14] P. R. Lorczak, A. K. Caglayan, and D. E. Eckhardt. A theoretical investigation of generalized voters for redundant systems. In *Proc. of the 19th Intl. Symposium on Fault-Tolerant Computing*, pages 444–451, June 1989.

[15] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200–209, April 1962.

[16] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

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

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

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

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

[21] M. J. Squair. Averages, voting and system robustness. Post to the author's blog (`criticaluncertainties.com`), January 2011.

[22] N. Storey. *Safety Critical Computer Systems*. Pearson Education Limited, Essex, England, 1996.

[23] A. Vargha and H. D. Delaney. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

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