Approach to component-based synthesis of fault-tolerant software

Behrooz Parhami
University of California, Santa Barbara, CA 93106, USA
Phone: +1 805 893 3211, Fax: +1 805 893 3262
parhami@ece.ucsb.edu

Keywords: components, design diversity, fault tolerance, multichannel computation, multiversion software, software reliability, safety, software reuse, weighted voting.

Received: June 6, 2001

N-version programming (NVP) and acceptance testing (AT) are established methods for obtaining highly reliable results from imperfect software. In NVP, several program modules are executed independently and the final result is derived by voting on the module outputs. In AT (as embodied, for example, in the recovery-block construct), outputs of a program module are subjected to an acceptance test and in the event of failing the test, alternate modules are invoked, until a module produces results that pass the test. Various symmetric combinations of NVP and AT techniques have also been suggested. We have found that a more general view, allowing the insertion of ATs at arbitrary points within a suitably constructed multichannel computation graph can lead to higher reliability and/or greater cost-effectiveness compared to the previously envisaged hybrid schemes such as consensus recovery blocks, recoverable N-version blocks, and N-self-checking programs. Accordingly, we introduce MTV graphs, and their simplified data-driven version called DD-MTV graphs, as component-based frameworks for the creation, representation, and analysis of hybrid NVP-AT schemes. MTV graphs model variations in fault-tolerant software architectures built of computation module (M), acceptance test (T), and voter (V) components. Following the definition of (DD-)MTV graphs, we present several examples of hybrid NVP-AT schemes, as instances of fault-tolerant software based on our component-based approach, and quantify the resulting reliability improvements. We show, for example, that certain, somewhat asymmetric, combinations of M, T, and V components lead to higher reliabilities and/or lower cost than previously proposed symmetric arrangements. We conclude that our component-based approach facilitates design space exploration for fault-tolerant software and leads to reliability improvements due to the double effect of architectural optimization and component refinement afforded by reuse.

1 Introduction

Applications of highly dependable computer systems are no longer limited to exotic space exploration and defense systems. A multitude of information and control systems in avionics, transportation, transaction processing, and process monitoring also rely on existence of ultrareliable computational resources [32], [43]. With the continually increasing complexity of hardware and software modules, and the attendant impossibility of implementing perfect (defect- or fault-free) components, the use of multi-channel computations with *design diversity* [2], [16], [17], [23], [51] has emerged as a practical and cost-effective approach.

Diverse multichannel computations take advantage of the property that, with adequate testing, malfunctions caused by residual design defects are rare and occur only for highly unusual sets of input conditions. It is thus likely that malfunctions of independently constructed modules, based on the same initial specifications, occur for different input states. This design diversity approach has been found useful for hardware subsystems [45] and for data [1] as well, but its primary application area is in constructing highly reliable software systems based on one of two distinct paradigms: Voting on multiple versions and acceptance testing of results [13], [24].

Following the success of hardware and data replication methods in tolerating physical faults in computing systems, the use of N-Version Programming (NVP) was proposed to allow tolerance of software design flaws [3], [8]. In NVP, several program modules are executed independently and the final result is obtained by voting on the module results. Voting, as used here, covers a wide variety of techniques in terms of sophistication, flexibility, and computational complexity [15], [30], [33] and need not be implemented through simple matching and majority rule. Several other terms, such as "consensus" [3] and "adjudication" [9] have been used to describe the decision process that computes an output based on possibly inexact or incomplete results provided by multiple redundant modules.

An important objection to NVP is that independence of design flaws in multiple versions cannot be guaranteed and that commonly used specification and software design techniques may lead to related faults in independently designed versions [21]. Such related faults may cause identical or similar errors and thus lead to an incorrect voter output. However, this is a criticism of dependability analyses for NVP and the attendant quantitative claims of reliability improvement. The usefulness of the NVP approach has never been in doubt. In some contexts, diversity is more readily ensured [35]. Also, attempts have been made to model the effects of correlated failures (e.g., [12], [25], [27]).

The technique of acceptance testing, e.g. as proposed in the recovery-block scheme [38], [39] is also based on design diversity. An acceptance test is an application-dependent routine that either accepts a result or declares it incorrect/suspect. Our confidence in an "accepted" result being actually "correct" depends on the thoroughness (coverage) of the acceptance test and its own reliability. ATs come in many different forms; from simple reasonableness checks to more complex, high-coverage validators. Assuming that the correctness of module result can be judged fairly accurately by applying an AT, alternate modules can be invoked sequentially in some specified order until one has produced a result passing the AT. This result is taken to be correct.

The main problem with redundancy techniques that rely solely on AT for validating a result is the difficulty of designing good acceptance tests that are both simple and thorough. Certain computations are easily checked through mathematical properties that relate the outputs to the inputs or by way of inverse computations where they exist [6]. In a great majority of cases, however, the only sure way of checking result validity with high confidence is to simply recompute using algorithms and/or hardware with diverse designs or operational characteristics.

Reliability evaluation for NVP and AT schemes has been presented in [5], [11], [23], [37], [41], [50], [52]. Linguistic constructs for describing adjudication and exception handling schemes for multi-version programs have been proposed in [26]. Kim et al [18], [19], [20] have studied architectures for, and design issues pertaining to, implementing a modified form of the recovery block scheme in a distributed environment. The resulting distributed recovery block (DRB) scheme uses concurrent execution of *try blocks* to allow fast forward recovery. Dugan and Lyu [10] discuss the relative merits of DRB, NVP, and NSCP (see the next paragraph).

Several groups have tried to combine NVP and AT. One such attempt is consensus recovery blocks (CRB) in which n versions are executed and their results are compared [40]. If there is agreement between two or more versions, then their common result is assumed correct and used. Otherwise, the n disagreeing results are subjected to an AT is some prespecified order and the first to pass the test is taken as the correct output. Another case is recoverable N-version blocks (RNVB) in which ATs are run on module outputs and only those that pass are provided to the voter [14]. This approach has also been suggested in [5], [16] and, under the name "N self-checking programming" (NSCP), in [22], [24]. Conceptually related to the efforts above, but different in terms of components and implementation, is *processor-data-check method*, and its graph-theoretic formulation, in algorithm-based fault tolerance [4], [44], [36].

Clearly, these are just examples of the ways in which NVP and AT approaches can be combined. CRB essentially applies NVP (with relaxed 2-out-of-n voting) and AT schemes sequentially and one at a time. Because no AT is applied in the case of, say, two agreeing results, there is some chance of an erroneous output being propagated. Furthermore, a common AT is assumed and design diversity is not applied to the AT. Both RNVB and NSCP approaches envisage applying ATs uniformly to all versions. This leads to an increase in complexity since ATs may essentially be duplicates of computational modules. We have found that a more general view, allowing the insertion of ATs at arbitrary points within a suitably constructed multichannel computation graph can lead to higher reliability and/or greater cost-effectiveness compared to the previously envisaged hybrid schemes such as consensus recovery blocks, recoverable N-version blocks, and N-self-checking programs.

Accordingly, we introduce MTV graphs, and their simplified data-driven version called DD-MTV graphs, as component-based frameworks for the creation, representation, and analysis of hybrid NVP-AT schemes. MTV graphs model variations in fault-tolerant software architectures built of computation module (M), acceptance test (T), and voter (V) components [34]. Following the definition of (DD-)MTV graphs, we present several examples of hybrid NVP-AT schemes, as instances of fault-tolerant software architectures developed based on our component-based approach, and quantify the reliability improvements they achieve. We show, for example, that certain, somewhat asymmetric, combinations of M, T, and V components can lead to higher reliabilities than previously proposed symmetric arrangements having comparable or higher complexities. We conclude that our component-based approach facilitates the exploration of the design space for fault-tolerant software and leads to reliability improvements due to the double effect of architectural optimization and component refinement afforded by reuse.

The rest of this paper is organized as follows. Section 2 contains basic definitions and assumptions as well as examples of a more general hybrid NVP-AT schemes in order to motivate the subsequent discussion. Sections 3 and 4 analyze NVP schemes in which 1 or k of the n versions, respectively, have been replaced by ATs. Section 5 deals with an example of more general combining schemes. Section 6 examines the effect of correlated failures. Conclusions and directions for further research appear in Section 7.

2 Terminology and assumptions

The question that we have set out to answer is how to combine the techniques of NVP and AT in an optimal way in order to achieve the best possible results. More specifically, our ultimate goal is to be able to combine diverse components (software modules, acceptance tests, voting algorithms) in a systematic way in order to maximize the correctness probability of the output with a given overall complexity or to achieve a desired correctness probability with minimal cost.

Unfortunately, in view of difficulties in estimating reliability and cost parameters, except in very limited cases [42], these problems are currently intractable when posed in their full generality. So, in this initial study, we endeavor to obtain results for a rather limited set of more specific questions with several simplifying assumptions.

Our hope is that with further research, the domain can be broadened and the assumptions gradually relaxed.

Here are our main objectives for this paper:

1. Demonstrate that certain novel, somewhat asymmetric, arrangements of modules, acceptance tests, and voters could be more reliable than earlier proposals.

2. Find optimality results in certain special cases; e.g., when the arrangement contains a single acceptance test or only one level of voting.

3. Provide examples of how the models and techniques used to analyze the special cases can be extended to deal with more complicated arrangements.

4. Generate interest in further systematic studies of the methodology, and tradeoff issues, for component-based synthesis of fault-tolerant software.

The following definitions and assumptions are needed in our discussions and analyses.

2.1. Definition – *MTV graph*: An MTV (Module-Test-Voter) graph is a directed acyclic graph with one "In", one "Out", and possibly one "Error" node, plus any number of nodes of three other types: Modules (M), acceptance tests (T), and voters (V).

M: Computes a result based on its inputs and sends it to a T or V node or to Out.

T: Accepts its input and forwards it or rejects it and activates some M or T nodes.

V: Forwards the result of weighted plurality voting or activates some M or T nodes.

The inner workings of M and T components are application-dependent. We make no assumption about these parts except that they have known, statistically independent reliability parameters (see 2.2-2.4). Voter components are more formally described in Def. 2.5. The nodes are connected by directed edges representing data transfers and controls. In diagrams, forwarding of data (the "P" output of an AT or the voting result from a V node) is represented by solid edges while activation controls (the "F" output of an AT or an indecisive voting outcome) are represented by dotted edges. ♦

Figures 1 and 2 contain examples of MTV graphs whose full meanings will become clear following the introduction of several more assumptions and definitions.

2.2. Assumption – *Reliability parameters of computation modules*: Each computation module M_i produces a result which is correct with fixed probability q_i and incorrect with fixed probability p_i, uniformly over its input space. In case $q_i + p_i < 1$, the module may be viewed as (partially) self-checking or fail-safe, abstaining from producing any result with probability $1 - q_i - p_i$. In the rest of this paper, we assume $q_i + p_i = 1$. ♦

Assumption 2.2 is controversial in that it is very difficult, if not impossible, to accurately estimate the reliability q_i of a software module [7]. We justify this assumption by noting that any system-level reliability analysis must be based on the reliability parameters of the components used. Accurate reliability estimates will be available with greater ease as we gain experience with the design and use of multiversion software. A Component-based strategy is helpful in this regard because component reuse fosters the gathering of more accurate reliability and performance data. Additionally, when comparing various arrangements of modules and tests, occasionally we obtain results indicating that one scheme is better than another for a wide range of parameter values or even for all values of a certain p_i. Hence, such comparative evaluations are less sensitive to, or totally independent of, the availability of accurate estimates for the p_is. The comments above apply to Assumption 2.3 as well.

2.3. Assumption – *Reliability parameters of acceptance tests*: A correct result passes an acceptance test T_i (outgoing edge labeled "P" is taken) with fixed probability q'_i and fails it (outgoing edge "F" is taken) with fixed probability p'_i, uniformly over the space of correct inputs. Similarly, T_i rejects an incorrect result with fixed probability q''_i and accepts it with fixed probability p''_i, uniformly over the space of incorrect inputs (note that in all definitions, q_is are close to 1 whereas p_is are near 0). When $q'_i + p'_i < 1$ or $q''_i + p''_i < 1$, the AT is viewed as (partially) self-checking or fail-safe, abstaining from judging an input with probability $1 - q'_i - p'_i$ for correct inputs and $1 - q''_i - p''_i$ for incorrect ones. Henceforth, we assume $q'_i + p'_i = q''_i + p''_i = 1$. ♦

The reason we do not start by assuming $q'_i = q''_i$ is that ATs behave asymmetrically with respect to correct and incorrect inputs. An AT that is itself defect-free, always accepts a correct input. Thus, p'_i is typically small and is related to the probability of a defect in the AT design. On the other hand, even a defect-free AT may accept an incorrect input due to imperfections in the testing algorithm (imperfect coverage). Hence, p''_i lumps together two sources of errors: imperfect coverage and defective design. We typically have $p''_i > p'_i$ (perhaps even, $p''_i >> p'_i$) for simple, low-complexity ATs. For a more comprehensive AT (e.g., one that duplicates the computation and decides by comparing the two values), coverage can be very high or even perfect. In such cases, q'_i and q''_i are comparable, though not necessarily equal.

2.4. Assumption – *s-independence of module and AT failures*: Each M and each AT fails s-independently of other Ms and ATs, unless otherwise noted. Hence, the probability of k modules M_i ($1 \le i \le k$) coincidentally producing erroneous results is $\Pi_{i\in[1,k]} p_i$. ♦

Assumption 2.4 is perhaps our most important assumption and the one most likely to be criticized. So let us try to justify it briefly. As noted in the introduction, the assumption of failure independence for multiversion software has been scrutinized and questioned from early on [21]. We think that these criticisms are valid and must be considered very seriously when trying to compute absolute reliability values for multichannel computations. However, the problem is much less serious for the types of analyses presented in this paper. Here, we try to determine if one scheme offers reliability improvement over another. Intuitively, since dependent failures are likely to affect the reliabilities of both schemes being compared, we can have a higher confidence in such relative figures of merit than in absolute reliabilities. We relax this independence assumption in Section 6 in order to validate, in part, this intuition. More work is clearly needed in this direction.

2.5. Definition – *Weighted plurality voter*: Given n input data objects $x_1, x_2, \ldots, x_n$, with associated nonnegative real votes (weights) $v_1, v_2, \ldots, v_n$, a V node computes the output object y and its vote w such that y is "supported by" a number of input data objects with votes totalling w and no other y' is supported by inputs having more votes. If $w \le (\sum_{i\in[1,n]} v_i)/2$, then the outcome y may be nonunique. In such cases, an erroneous voter output will be pessimistically assumed. The output weight w selects one of the outgoing voter edges along which y or an activation signal must be sent (see, e.g., Fig. 1). When input votes are not explicitly specified, it is assumed that $v_1 = v_2 = \ldots = v_n = 1$. Various definitions of the term "supported by" lead to different voting schemes such as exact, inexact, and approval voting (e.g., with exact voting, an input object x_i supports y iff $x_i = y$). These variations, although important, are beyond the scope of this paper [30], [33]. ♦

2.6. Assumption – *Perfect voters*: Voters are perfect and act instantaneously. This assumption is reasonable because voters are simpler than modules or ATs and are designed just once for use with many different modules and test types. They can be made highly reliable through careful design and extensive testing (much more so than what is reasonable for any single application or its associated ATs). ♦

Consider Fig. 1 in which 5VP and an alternative scheme with the same number of M/T modules, and thus with lower or equal complexity, are shown. T is an acceptance test with pass/fail (P/F) outcome. A result that passes T is simply forwarded to the output. When T rejects its input, it activates M_4. The voter in Fig. 1c can produce one of three mutually exclusive outputs: (1) No agreement, leading to activation of M_4, (2) agreement between two inputs, leading to the application of T on the agreed-upon result, and (3) agreement among all three inputs, yielding an acceptable output.

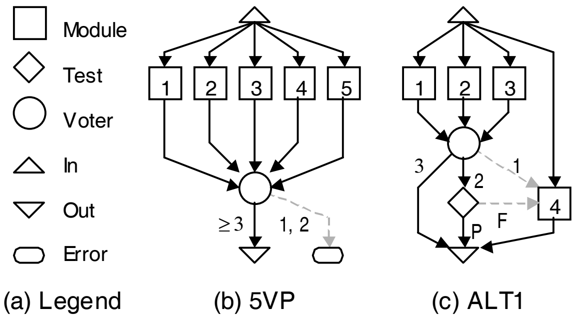

Fig. 1. Representations of 5VP and ALT1, an alternative scheme, as MTV graphs.

Figure 2 shows MTV graphs corresponding to two other hybrid NVP-AT schemes. These schemes have been proposed in the literature as alternatives to 3VP, although both imply greater cost than 3VP. Figure 2a represents a 3-channel RNVB or NSCP. Each of the three computation channels is made self-checking through the insertion of an AT after the computation module. The V node in Fig. 2a is a weighted plurality voter for which each input weight is set to 0 or 1, based on the outcome of the associated AT.

Figure 2b represents a 3-channel CRB. If the voter observes agreement between two or all three of its inputs, the agreed-upon result is forwarded to the output. On the other hand, when there is no agreement, results from the computation channels are successively subjected to an AT and the first to pass the test is forwarded to output. The ATs in Fig. 2b are labeled T_1, T_2, and T_3, but they may all represent the same test. The advantage of this scheme over 3VP is that it is guaranteed to produce the correct result whenever 3VP would produce the correct result. Additionally, the added AT mechanism may salvage a correct result from disagreeing modules.

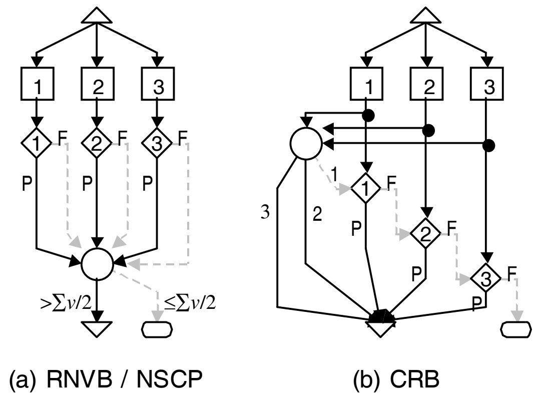

Fig. 2. MTV graphs representing two different 3-channel hybrid NVP-AT schemes.

Returning now to the initial example, the 5VP scheme of Fig. 1b can tolerate up to two module failures, but can produce an incorrect result with some triple failures. In fact, any worst-case analysis must assume that 5VP fails when there are three module failures. The alternative scheme shown in Fig. 1c also tolerates any two failures in the M/T nodes. This claim is proven by considering the following four cases which exhaust all possible double failures.

Case 1 – Two failures in $\{M_1, M_2, M_3\}$: T and M_4 are fault-free. If the two faulty modules produce mutually supportive results, then the error is caught by T and M_4 is executed. Otherwise, M_4 is executed directly. Either way, the correct result is produced.

Case 2 – Two failures in $\{T, M_4\}$: M_1, M_2, and M_3 are fault-free and produce pairwise mutually supportive results. The voter outputs is the correct result.

Case 3 – One failure in $\{M_1, M_2, M_3\}$ and one in T: Because M_4 is fault-free, the output is correct whether or not T accepts the correct result received from the voter.

Case 4 – One failure in $\{M_1, M_2, M_3\}$ and one in M_4: T is fault-free and accepts the voter's majority result.

Triple failures can lead to an incorrect output for the alternative scheme in a manner similar to 5VP. For example, if M_1, M_2, and M_3 are faulty but produce incorrect pairwise mutually supportive results, an incorrect value will be output. Failure of M_1, T, and M_4 also can lead to incorrect output. A side benefit of the alternative scheme of Fig. 1c is that whereas all five modules must run to completion in 5VP, the alternative scheme rarely needs to execute M_4.

Let us now analyze the two schemes of Fig. 1 with respect to reliability (probability of producing a correct result). Assuming $p_1 = p_2 = p_3 = p_4 = p = 1 - q$:

$$Q_{5VP} = q^5 + 5q^4p + 10q^3p^2 = q^2(1 + 2p + 3p^2 - 6p^3)$$

$$\begin{aligned} Q_{ALT1} &= q^3 + 3q^2p(1 - p'p) + 3qp^2q''q \\ &= q^2[1 + 2p + 3p^2 - 3p^2(p' + p'')] \end{aligned}$$

The equation for the reliability of the alternate configuration in Fig. 1c, Q_{ALT1}, is derived as follows. The first term, q^3, is the probability of having no fault in $\{M_1, M_2, M_3\}$. The second term, $3q^2p(1 - p'p)$, covers the event of having a single fault in $\{M_1, M_2, M_3\}$, in which case correctness of the result is guaranteed unless T rejects the correct majority result *and* M_4 is faulty. The third term, $3qp^2q''q$, corresponds to having two faults in $\{M_1, M_2, M_3\}$. In this event, the worst case is the agreement of the two faulty modules because it leads to the requirement for T to reject the incorrect majority result (probability q'') *and* for M_4 to be fault-free (probability q) to obtain the correct result. If the two faulty modules in $\{M_1, M_2, M_3\}$ disagree with each other and with the correct result, then the only requirement for producing the correct output is for M_4 to be fault-free.

Comparing the two expressions above, we find that $Q_{\text{ALT1}} > Q_{\text{5VP}}$ iff $p' + p'' < 2p$. In the special case of $p' = p'' = s$, the alternate scheme ALT1 offers reliability improvement over 5VP iff $s < p$; i.e., the alternate scheme is better if the acceptance test component T is more reliable than each module M_i. In practice, T can often be made simpler, and thus more reliable, than M_i. This may be due to the inherent simplicity of verifying the result's correctness (e.g., by means of a mathematical relationship or an inverse computation) or by virtue of T using extra information ("certification trail") provided by the computation modules [46], [47], [48].

The examples above were intended to demonstrate the power of MTV graphs as representation and analysis tools for multichannel computations. In particular, it was shown that previously proposed hybrid NVP-AT approaches can be represented as simple MTV graphs and that these graphs can also represent more general hybrid schemes that have not been dealt with in the past and that can potentially offer higher reliability and/or lower overall complexity. Next, we define a modified form of MTV graphs in order to simplify the discussion in the remainder of this paper.

2.7. Definition – *Data-driven MTV (DD-MTV) graph*: A DD-MTV graph is a modified MTV graph with no Error node and only single-output M, T, and V nodes.

M_i: Attaches the weight w_i to its output y_i.

T_i: Modifies the weight w of y to $w + a_i(w)$ or $w - r_i(w)$ upon acceptance/rejection.

V_i: Produces data object y of weight w from its inputs, as detailed in Def. 2.5.

The Error node is not needed because error can be indicated by a subset of possible weights (low values) for the final result. The elimination of control edges and the resultant graph's uniformity simplifies the enumeration and analysis of various alternatives, while still retaining the power to accurately model most hybrid schemes. The weight augmentation and reduction functions, $a_i(w)$ and $r_i(w)$, used to adjust the weight of an accepted and rejected input, respectively, are nonnegative functions to be determined (see Assumption 2.9). ♦

Figure 3 depicts several examples of DD-MTV graphs, each having six M/T nodes. These can be viewed as lower-complexity alternatives to 6VP. The examples in Fig. 3 clearly show the wide variety of multichannel computational arrangements that can be modeled easily by DD-MTV graphs [29].

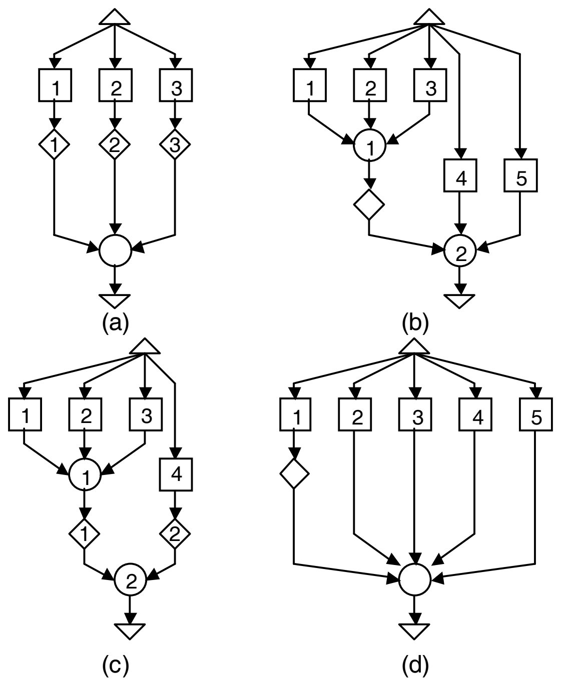

Fig. 3. DD-MTV graphs with six M/T nodes.

2.8. Assumption – *Uniformity of modules and ATs*: In the rest of this paper, modules will be assumed to have identical reliability, complexity, and execution-time parameters. Thus, the subscript i will be omitted from parameters such as p_i, q_i and the weight $w = 1$ is attached to all module outputs. Similarly, ATs will be uniformly treated by eliminating the subscript i from their respective parameters such as p'_i and p''_i. ATs will be taken to have perfect coverage and lower or equal complexity compared to modules, thus leading to the assumptions $p' \le p$ and $p'' \le p$. ♦

2.9. Assumption – *Weight augmentation/reduction functions for ATs*: Selection of appropriate weight augmentation and reduction functions, $a(w)$ and $r(w)$, can have important effects on the overall reliability of the system modeled by a particular DD-MTV graph. In this paper, we assume $a(w) = r(w) = 1$. These simple constant functions can be intuitively justified when ATs have near-perfect coverage and are of comparable complexity to modules, and they have worked well in practice. However, an extensive study of techniques for optimally choosing these functions is required. ♦

The stage is now set for a more detailed examination of certain classes of DD-MTV graphs and the systems they model. Before that, we recap the abbreviations used and introduce some needed notation.

2.10. Notation and nomenclature – The following is a list of symbols and abbreviations used in the paper:

AT	Acceptance Test(ing)
CRB	Consensus Recovery Block (Fig. 2b)
$C_{x,y}$	Binomial coefficient = $x!/[y!(x-y)!]$
DD-MTV	Data-Driven MTV graph (Def. 2.7)
M	Module; node in (DD-)MTV graph
MTV	Module-Test-Voter graph (Def. 2.1)
NSCP	N-Self-Checking Program (Fig. 2a)
nVP	n-Version Program(ming); e.g., 3VP
P	Failure probability = unreliability = $1 - Q$

Q (Q_X)	Reliability (of system or configuration X)
$R_{k,m}$	Reliability of k-out-of-m system
RNVB	Recoverable N-Version Block (Fig. 2a)
T	Test (AT) node in (DD-)MTV graph
V	Voter node in (DD-)MTV graph ♦

3 Replacing one version with an AT

When one module in 3VP is replaced by an acceptance test, a recovery block scheme with one alternate is obtained (Fig. 4a). Note that Fig. 4a is a correct model of recovery block scheme as far as reliability estimation is concerned. The fact that M_1 and M_2 appear to be running in parallel rather than M_2 following M_1, and then only in case T rejects M_1's result, is irrelevant to the reliability calculation. In this section, we generalize this notion by analyzing the effect of replacing one module in nVP with an AT (Fig. 4b). Reliability expressions for 3VP and the alternate scheme ALT2 of Fig. 4a are as follows:

$$Q_{3VP} = q^3 + 3q^2p = q(1 + p - 2p^2)$$

$$Q_{ALT2} = qq' + qp'q + pq''q = q[1 + p - p(p' + p'')]$$

The equation for the reliability of the alternate configuration in Fig. 4a, Q_{ALT2}, is derived as follows. The first term, qq', is the probability of M_1 producing the correct result *and* T accepting it. The second term, $qp'q$, covers the event of M_1 producing the correct result, T rejecting it, *and* M_2 getting the correct result. The third term, $pq''q$, corresponds to M_1 producing an incorrect result, T catching the error, *and* M_2 being fault-free. Comparing the expressions above, we find that $Q_{ALT2} > Q_{3VP}$ iff $p' + p'' < 2p$. Hence the discussion preceding Def. 2.7 applies here as well. Figure 5 shows the unreliability $P = 1 - Q$ of 3VP and ALT2 when $p' = p''$.

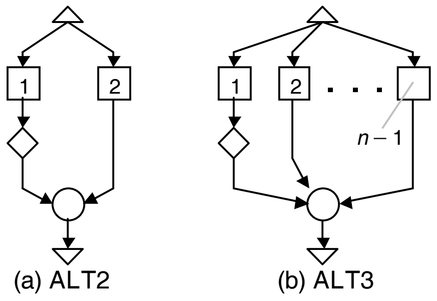

Fig. 4. Replacing one module with an acceptance test in 3VP and nVP.

It is relatively straightforward to generalize the analysis above to the comparison of nVP and the alternative scheme ALT3 depicted in Fig. 4b. However, we first need some notation. Let $R_{k,m}$ be the reliability of a homogeneous k-out-of-m system in which each module fails with probability p (for brevity, the parameter p is not explicitly shown).

$$R_{k,m} = \sum_{j\in[k,m]} C_{m,j}\, q^j p^{m-j}$$

where $C_{m,j}$ is the binomial coefficient. $R_{k,m}$ is defined to be 0 for $k > m$ and 1 for $k \le 0$. We now write reliability equations for nVP and the ALT3 scheme of Fig. 4b as follows. To simplify the formulas, let $h = \lfloor n/2 \rfloor$. Each of the following expressions is written by considering the four possible cases with respect to the presence of faults in two modules or in one module and its associated AT and for each case figuring out how many of the remaining $n - 2$ modules must be fault-free in order to guarantee a correct result.

$$Q_{nVP} = R_{h+1,n} = q^2R_{h-1,n-2} + 2pqR_{h,n-2} + p^2R_{h+1,n-2}$$

$$Q_{ALT3} = qq'R_{h-1,n-2} + qp'R_{h,n-2} + pq''R_{h,n-2} + pp''R_{h+1,n-2}$$

To compare these reliabilities, let us compute their difference $\Delta Q = Q_{ALT3} - Q_{nVP}$:

$$\begin{aligned}\Delta Q &= q(p - p')R_{h-1,n-2} + [p(p - p'') - q(p - p')]R_{h,n-2} \\ &\quad - p(p - p'')R_{h+1,n-2} \\ &= q(p - p')[R_{h-1,n-2} - R_{h,n-2}] \\ &\quad + p(p - p'')[R_{h,n-2} - R_{h+1,n-2}]\end{aligned}$$

Because each of the two terms within the square brackets is positive, a sufficient condition for reliability improvement over nVP is immediately obtained as $max(p', p'') < p$, which always holds by Assumption 2.8.

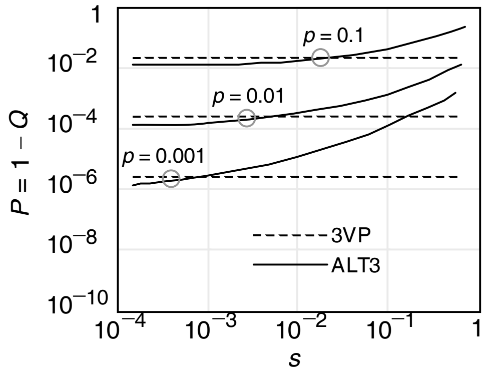

Fig. 5. Unreliability $P = 1 - Q$ of 3VP and ALT2, assuming $p' = p'' = s$.

To continue the analysis, we note that:

$$\begin{aligned}R_{k-1,m} - R_{k,m} &= C_{m,k-1}q^{k-1}p^{m-k+1} \\ &= m!q^{k-1}p^{m-k+1}/[(k - 1)!(m - k + 1)!]\end{aligned}$$

Hence, we can rewrite $\Delta Q = Q_{ALT3} - Q_{nVP}$ as:

$$\begin{aligned}\Delta Q &= q(p - p')(n - 2)!q^{h-1}p^{n-h-1}/[(h - 1)!(n - h - 1)!] \\ &\quad + p(p - p'')(n - 2)!q^hp^{n-h-2}/[h!(n - h - 2)!] \\ &= \{(n - 2)!/[h!(n - h - 1)!]\}\, q^hp^{n-h-1} \\ &\quad \times [(n - 1)p - hp' - (n - h - 1)p'']\end{aligned}$$

Therefore, the sign of $\Delta Q = Q_{ALT3} - Q_{nVP}$ depends on the sign of the last expression within square brackets. For n odd, we have $h = (n - 1)/2$ and $\Delta Q > 0$ iff $p' + p'' < 2p$. For n even, we have $h = n/2$ and $\Delta Q > 0$ iff $(n/2 - 1)(p' + p'') + p' < (n - 1)p$. In this latter case, p' is somewhat more important than p''. As an example, for $n = 6$, we must have $3p' + 2p'' < 5p$ if the alternate with one AT (Fig. 3d) is to be more reliable than 6VP.

4 Replacing k versions with ATs

We now consider the case where k of the n modules are removed ($k \le n/2$) and replaced by ATs following k of the remaining $n - k$ modules. As shown in the MTV graph of Fig. 6a, k branches with modules $M_1, M_2, \ldots, M_k$ include acceptance tests $T_1, T_2, \ldots, T_k$ and $n - 2k$ branches have just modules (indexed from k+1 to n–k).

As in Section 3, let $R_{k,m}$ be the reliability of a homogeneous k-out-of-m system in which each module fails with probability p and let $h = \lfloor n/2 \rfloor$ for notational convenience. We can then write:

$$Q_{n\text{VP}} = R_{h+1,n} = \sum_{i\in[h+1,\,n]} C_{n,i}\, q^i p^{n-i}$$

$$Q_{\text{ALT4}} = \sum_{i\in[0,\,k]}\sum_{j\in[0,\,k-i]}\{ C_{k,i}C_{k-i,j}(pp'')^i(pq''+qp')^j \times (qq')^{k-i-j} R_{h+2i+j-2k+1,n-2k}\}$$

The reliability expression Q_{ALT4} for the alternate configuration is derived as follows. Let of the k branches containing ATs, i have faults in both the module and in the AT, j have a fault in either the module or the AT but not both, and $k - i - j$ be fault-free. The i branches with double faults all potentially produce incorrect results with weight 2. The j branches containing single faults produce results with weight 0, whether the fault is in M or in T. Finally, the $k - i - j$ fault-free branches produce correct results with weight 2. If c of the remaining $n - 2k$ modules produce correct results, the following condition must be met for the output to be guaranteed correct:

$$c + 2(k - i - j) > (n - 2k - c) + 2i$$
$$\Rightarrow \quad c \geq h + 2i + j - 2k + 1$$

This justifies the term $R_{h+2i+j-2k+1,n-2k}$ in the expression for Q_{ALT4}. The remaining terms are the probabilities of the indicated number of faults raised to appropriate powers. For example, the probability that M is faulty but T catches the error or M is fault-free but T rejects its output is $pq'' + qp' = p + p' - pp' - pp''$.

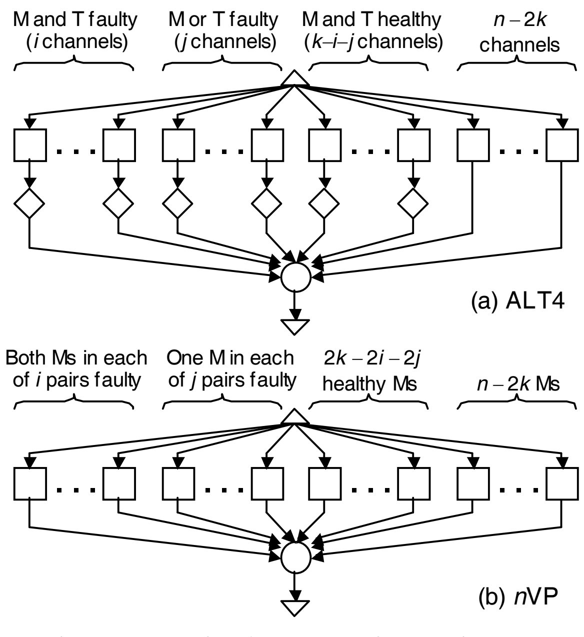
M and T faulty (i channels)
M or T faulty (j channels)
M and T healthy (k–i–j channels)
n – 2k channels
(a) ALT4
Both Ms in each of i pairs faulty
One M in each of j pairs faulty
2k – 2i – 2j healthy Ms
n – 2k Ms
(b) nVP

Fig. 6. Replacing k modules with ATs in nVP and the notation for reliability analysis.

As written, the expressions for $Q_{n\text{VP}}$ and Q_{ALT4} are difficult to compare without resorting to numerical calculation. To facilitate comparison, we rewrite the expression for $Q_{n\text{VP}}$ in the following way. We divide the set of n modules into k module pairs plus $n - 2k$ individual modules, as shown in Fig. 6b. Each of the k pairs can have 2, 1, or 0 faulty modules. Let i, j, and $k - i - j$ be the number of such pairs, respectively. The k module pairs contribute incorrect results with total weights of up to $2i + j$ and correct results with total weights of at least $2k - 2i - j$. Again, if the remaining $n - 2k$ modules produce c correct results and $n - 2k - c$ incorrect ones, we must have $c + 2k - 2i - j > (n - 2k - c) + 2i + j$, or $c \geq h + 2i + j - 2k + 1$, to guarantee a correct output. The probabilities of having 2, 1, ore 0 faulty modules in a pair of modules are p^2, $2pq$, and q^2, respectively. Thus:

$$Q_{n\text{VP}} = \sum_{i\in[0,\,k]}\sum_{j\in[0,\,k-i]}\{ C_{k,i}C_{k-i,j}(p^2)^i(2pq)^j \times (q^2)^{k-i-j} R_{h+2i+j-2k+1,n-2k}$$

Comparing the corresponding ij terms in the expression for Q_{ALT4} to the above expression for $Q_{n\text{VP}}$ provides some insight but no general conclusion. For example, for $p' = p'' = s$, corresponding terms become identical and the two schemes are equivalent with respect to reliability. For $p' = p'' = s$, the ij term in Q_{ALT4} divided by the ij term in $Q_{n\text{VP}}$ yields the ratio:

$$(s/p)^i[(p + s - 2ps)/(2pq)]^j[(1 - s)/q]^{k-i-j}$$

For particular values of s and p satisfying $s < p$, the first and the second term above are always less than 1 while the third term is always greater than 1. Hence, the ratio can be less than or greater than 1 depending on the values of i and j and no conclusion can be drawn based on this term-by-term comparison.

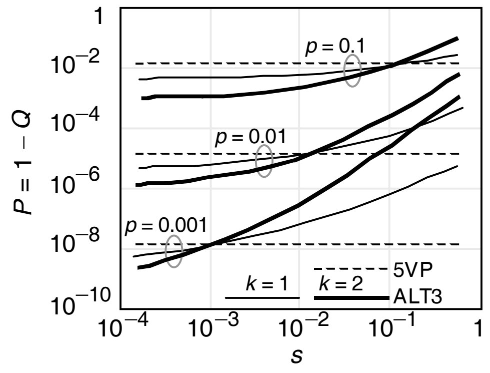

Fig. 7. Unreliability P=1–Q of 5VP and ALT4 with k = 1 or 2, assuming $p' = p'' = s$.

For n = 3, the only acceptable value for k is 1 and Fig. 5 depicts the corresponding changes in the unreliability $P = 1 - Q$. To note the effect of changing k, the expressions for $Q_{n\text{VP}}$ and Q_{ALT4} have been evaluated for n = 5, with k = 1 or 2, and p = 0.1, 0.01, or 0.001, assuming $p' = p'' = s$. Figure 7 depicts the resulting unreliabilities as functions of s.

As expected, the unreliabilities $P_{n\text{VP}}$ and P_{ALT4} are identical for $s = p$ (see the crossover points in Fig. 7). The 5VP scheme is uniformly better for $s > p$. In the case of $s < p$, both alternates are uniformly better than 5VP and the alternate with k = 2 is better than that with k = 1. It is worth noting that the improvement in reliability achieved for $s < p$ is smaller than the degradation suffered for $s > p$, particularly for larger k. Therefore, modules must be replaced with ATs only if the condition $s < p$ is reasonably certain.

5 More general schemes

As seen from the examples given in Fig. 3, DD-MTV graphs and the systems they model can be composed in many different ways. The systems discussed and analyzed in Sections 3 and 4 all involve a single level of voting. In this section, we discuss a system involving two levels of voting as an example of more general schemes. It is hoped that several other arrangements will be covered in the continuation of this research.

Consider the MTV graph ALT5 depicted in Fig. 8 as an alternative to nVP. The result of k-way voting on k of the modules is given to T and is then combined with the results of $n - k - 1$ modules through a second-level weighted voter. This may be viewed as a generalized recovery block scheme in which the primary computation consists of a k-way voted block and the alternate consists of an $(n - k - 1)$-way parallel block. In an actual implementation, modules in the alternate block may be executed sequentially until sufficient votes are collected, given the outcome produced by the primary voting block. The reliability of ALT5 is:

$$\begin{aligned} Q_{\mathrm{ALT5}} = {} & \textstyle\sum_{i\in[0,\lfloor k/2\rfloor]}\{C_{k,i}q^ip^{k-i} \\ & \quad \times [q''R_{\lfloor(n-i)2\rfloor,n-k-1} + p''R_{\lfloor(n-i)2\rfloor+1,n-k-1}]\} \\ & + \textstyle\sum_{i\in[\lfloor k/2\rfloor+1,\,k]}\{C_{k,i}q^ip^{k-i} \\ & \quad \times [q'R_{\lfloor(n-k-i)2\rfloor,n-k-1} + p'R_{\lfloor(n-k-i)2\rfloor+1,n-k-1}]\} \end{aligned}$$

The reliability expression Q_{ALT5} is derived as follows. Let there be i correct results among the k channels of the primary voting block. This event has the probability $C_{k,i}q^ip^{k-i}$. Now if $i \le \lfloor k/2\rfloor$, the plurality voting result must be assumed incorrect in the worst case. If T rejects this incorrect result (probability q''), its weight is decreased to $i - 1$ and a correct final output will be produced as long as at least $\lfloor(n - i)/2\rfloor$ of the remaining $n - k - 1$ modules are fault-free. On the other hand, if T erroneously accepts the incorrect result (probability p''), thus increasing its weight to $i + 1$, then at least $\lfloor(n - i)/2\rfloor + 1$ of the remaining $n - k - 1$ modules must be fault-free for the final result to be guaranteed correct. Recall that $R_{j,m} = 0$ for $j > m$. Similarly, if $i \ge \lfloor k/2\rfloor + 1$, then the voter output is correct and has a weight of i. A similar argument justifies the second half of the expression for Q_{ALT5}.

To compare Q_{ALT5} to $Q_{n\mathrm{VP}}$, we divide the n modules into three groups of k, 1, and $n - k - 1$ modules. If i is the number of fault-free modules in the first group, then:

$$\begin{aligned} Q_{n\mathrm{VP}} = {} & \textstyle\sum_{i\in[0,\,k]}\{C_{k,i}q^ip^{k-i} \\ & \quad \times [qR_{\lfloor(n-k-i)/2\rfloor,n-k-1} + pR_{\lceil(n-k-i)2\rceil+1,n-k-1}]\} \end{aligned}$$

The two terms within square brackets correspond to the case of the single module in the second group being fault-free (probability q) or faulty (probability p), respectively, leading to different requirements for the number of fault-free modules in the third group ($i + 1 + c > n - k - 1 - c$ in the first case and $i + c > n - k - 1 - c$ in the second, where c is the required number of fault-free modules in the third group). Again comparison of the expressions for Q_{ALT5} and $Q_{n\mathrm{VP}}$ leads to no general conclusion. For example, in the case of $p' = p'' = p$, we note that of the corresponding pairs of terms in the expressions for Q_{ALT5} and $Q_{n\mathrm{VP}}$, some are larger in Q_{ALT5} and others are larger in $Q_{n\mathrm{VP}}$.

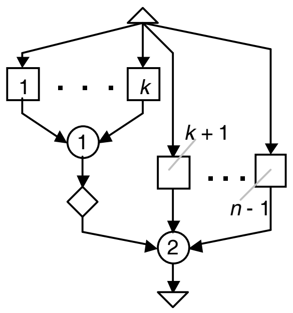

Fig. 8. A DD-MTV graph containing two levels of voting (ALT5).

To get a feel for the relative values of Q_{ALT5} and $Q_{n\mathrm{VP}}$ and conditions under which the alternative scheme offers a higher reliability than nVP, consider the special case depicted in Fig. 3b ($n = 6$, $k = 3$). The relevant reliability equations in this case are:

$$Q_{6\mathrm{VP}} = q^2(1 + 2p + 3p^2 - 16p^3 + 10p^4)$$

$$\begin{aligned} Q_{\mathrm{ALT5}}\{n{:}6, k{:}3\} = {} & q^2[1 + 2p + 3p^2 - 3p^3 \\ & - p^2(1 + 2p)p' - 3p^2(1 - p)p''] \end{aligned}$$

To compare these reliabilities, let us compute their difference $\Delta Q = Q_{\mathrm{ALT5}} - Q_{6\mathrm{VP}}$:

$$\Delta Q = q^2p^2[p(13 - 10p) - (1 + 2p)p' - 3(1 - p)p'']$$

Therefore, for the alternative scheme to be better than 6VP, we must have:

$$(1 + 2p)p' + 3(1 - p)p'' < p(13 - 10p)$$

Observe that p'' is more important than p' in that it is multiplied by a larger factor. In the special case of $p' = p'' = s$, The condition above becomes $s < p(13 - 10p)/(4 - p)$ or $s < 3p + p(1 - 7p)/(4 - p)$. Thus, for p reasonably small, reliability improvement is guaranteed as long as s is no larger than $3p$.

One cannot draw general conclusions on the basis of a single example, but it is interesting to pinpoint the cause of the reliability improvement in this special case. Both 6VP and the scheme depicted in Fig. 3b produce the correct result when at least four M/T nodes are fault-free. To see this in the case of Fig. 3b, consider the following five cases which exhaust all possible double failures.

Case 1 – Two failures in $\{M_1, M_2, M_3\}$: Fault-free T rejects the incorrect voter output, reducing its weight from 2 to 1. Correct output is produced because M_4 and M_5 are both fault-free.

Case 2 – One failure in $\{M_1, M_2, M_3\}$ and one in T: T rejects the correct voter output, reducing its weight from 2 to 1. The output would be correct even if M_4 or M_5 were faulty.

Case 3 – One failure in $\{M_1, M_2, M_3\}$ and one in $\{M_4, M_5\}$: T accepts the correct voter output, increasing its weight to 3. The output is independent of M_4 or M_5.

Case 4 – One failure in T and one in $\{M_4, M_5\}$: T rejects the correct voter output, reducing its weight from 3 to 2. The fault-free module in $\{M_4, M_5\}$ creates a correct majority.

Case 5 – Two failures in $\{M_4, M_5\}$: T is fault-free and accepts the unanimous voter output, increasing its weight from 3 to 4. M_4 and M_5 cannot affect this output.

Cases 2 and 3 above show that some triple failures are also tolerated by the alternative scheme; hence the improved reliability. In certain instances, as in Cases 3 and 5 above, the output of T obviates the need for executing M_4 or M_5. These cases correspond to up to one fault in $\{M_1, M_2, M_3\}$, with T fault-free, and have a probability of $q^2(1 + 2p)(1 - s)$.

One should note that if there were no AT between the two voting levels in Fig. 8, reliability would actually degrade compared to a single-level scheme with the same number of modules. The reason is that correct minority results in the first level are discarded whereas they may help establish a correct majority if combined with correct outputs from the remaining modules. So the AT is a key component in this multilevel voting configuration. Multilevel voting without some form of intermediate validation is simply not beneficial.

6 Dealing with correlated failures

General analysis of various hybrid redundancy schemes with correlated failures becomes significantly more complex. In this section, we present a simplified analysis based on a highly pessimistic view of correlated failures: that they affect a set of modules and ATs in the worst possible way, causing the modules to produce identical incorrect results and an AT to reject any correct result and to accept any incorrect result. We obtain lower bounds for the reliabilities of pure and hybrid schemes and show the bounds corresponding to certain hybrid schemes to be higher. However, this does not necessarily imply that the hybrid schemes are more reliable, because $a > b$, $a' > b'$, and $b > b'$ do not imply $a > a'$. On the other hand, reliability of a complex system can never be computed exactly and we usually settle for lower bound guarantees. From this viewpoint, a system for which the lower-bound, or guaranteed reliability level, is higher must be considered better.

In what follows, we compare nVP and ALT3 configurations (see Fig. 4b) with regard to correlated failures and show ALT3 to be superior. The comparison is based on combinatorial analysis. Admittedly, the application of this method would be cumbersome for more complicated configurations. However, our aim here is to validate, in part, the intuition that the possibility of correlated failures does not alter our earlier conclusions. The insight gained from this example analysis will help us understand why replacement of some modules with ATs improves the probability of obtaining a correct result under both statistically independent and correlated failure scenarios.

Because nVP and ALT3 differ only in the use and placement of M_1, M_n, and T, our model postulates the occurrence of correlated failures in c modules among $\{M_2, \ldots, M_{n-1}\}$ and includes probability parameters relating to how M_1, M_n, and/or T may be affected. The parameters β, β', σ, μ, τ, ν, ν', defined below, should be interpreted as "probability of *event*, given that c modules among $\{M_2, \ldots, M_{n-1}\}$ contain correlated/common failures". Nodes unaffected by correlated failures can still suffer from random failures, with corresponding parameters as defined in Section 2. The events associated with the conditional probabilities for nVP and ALT3 are:

NVP	β	Both M_1 and M_n are affected
	σ	Single module: M_1 or M_n is affected
	ν	Neither M_1 nor M_n is affected
ALT3	β'	Both M_1 and T are affected
	μ	M_1 is affected but T is not
	τ	T is affected but M_1 is not
	ν'	Neither M_1 nor T is affected

Clearly, we have $\beta + 2\sigma + \nu = \beta' + \mu + \tau + \nu' = 1$. Also, given that two modules are more similar than a module and an AT, the following might be considered reasonable assumptions:

$$\tau \le \sigma \le \mu \qquad \text{and} \qquad \beta' \le \beta \le \sigma \le \nu \le \nu'$$

These inequalities are essentially the crux of our comparative evaluation, in much the same way that the assumption $p' + p'' < 2p$ was essential in proving improvements with independent failures. One last item of notation: Because the reliability of a k-out-of-$(n - c - 2)$ system, $R_{k,n-c-2}$, is used repeatedly in the following analysis, we denote it by R_k for brevity. Recall that h was defined as $\lfloor n/2 \rfloor$.

We next derive upper bounds on the reliability reduction due to correlated failures in nVP and ALT3. The "≈" sign denotes proportionality rather than equality.

$$\begin{aligned}\Delta Q_{n\mathrm{VP}} &\approx \beta(1 - R_{h+1}) + 2\sigma[q(1 - R_h) + p(1 - R_{h+1})] \\ &\quad + \nu[q^2(1 - R_{h-1}) + 2pq(1 - R_h) + p^2(1 - R_{h+1})] \\ &= 1 - [(\beta+2\sigma p+\nu p^2)R_{h+1} + 2q(\sigma+\nu p)R_h + \nu q^2 R_{h-1}]\end{aligned}$$

$$\begin{aligned}\Delta Q_{\mathrm{ALT3}} &\approx \beta'(1 - R_{h+1}) + \mu[q''(1-R_h) + p''(1-R_{h+1})] \\ &\quad + \tau[q(1-R_h) + p(1-R_{h+1})] + \nu'[qq'(1 - R_{h-1}) \\ &\quad + p'q(1 - R_h) + pq''(1 - R_h) + pp''(1 - R_{h+1})] \\ &= 1 - [(\beta' + \mu p'' + \tau p + \nu' pp'')R_{h+1} \\ &\quad + (\mu q''+\tau q+\nu' p'q+\nu' pq'')R_h + \nu' qq' R_{h-1}]\end{aligned}$$

$$\begin{aligned}\Delta Q_{n\mathrm{VP}}-\Delta Q_{\mathrm{ALT3}} &\approx (\beta'-\beta+\mu p''+\tau p-2\sigma p+\nu' pp''-\nu p^2)R_{h+1} \\ &\quad + (\mu q''+\tau q-2\sigma q+\nu' p'q+\nu' pq'' R_h-2\nu pq)R_h \\ &\quad + q(\nu' q'-\nu q)R_{h-1}\end{aligned}$$

To simplify the expression for $\Delta Q_{n\mathrm{VP}} - \Delta Q_{\mathrm{ALT3}}$, note the following equalities:

$$R_h = R_{h+1} + C_{n-c-2,h}\, q^h p^{n-c-2-h}$$

$$R_{h-1} = R_{h+1} + C_{n-c-2,h}\, q^h p^{n-c-2-h} + C_{n-c-2,h-1}\, q^{h-1} p^{n-c-1-h}$$

Substituting the preceding in the expression for $\Delta Q_{n\mathrm{VP}} - \Delta Q_{\mathrm{ALT3}}$, the coefficient for R_{h+1} becomes 0. Dividing both sides by $(n - c - 2)!q^h p^{n-c-2-h}/[h!(n - c - 1 - h)!]$, yields:

$$\begin{aligned}\Delta Q_{n\mathrm{VP}} - \Delta Q_{\mathrm{ALT3}} &\approx (n-c-1-h)[q(\mu q''/q+\tau-2\sigma+\nu'-\nu) \\ &\quad + p(\nu' q''-\nu q)] + hp(\nu' q'-\nu q)\end{aligned}$$

Because by our assumptions both $\nu' q'' - \nu q$ and $\nu' q' - \nu q$ are nonnegative, a sufficient condition for the difference $\Delta Q_{n\mathrm{VP}} - \Delta Q_{\mathrm{ALT3}}$ to be nonnegative is to have $\mu q''/q + \tau - 2\sigma + \nu' - \nu \ge 0$.

$$\begin{aligned}\mu q''/q+\tau-2\sigma+\nu'-\nu &= \mu(q''-q)/q + (\mu+\tau+\nu') - (2\sigma+\nu) \\ &= \mu(q''-q)/q + (1-\beta') - (1-\beta) = \mu(q''-q)/q + \beta - \beta'\end{aligned}$$

This last expression is nonnegative by our assumptions; hence $\Delta Q_{n\mathrm{VP}} \ge \Delta Q_{\mathrm{ALT3}}$ and the conclusion that correlated failures have a less serious effect on ALT3 than on nVP.

7 Conclusion

The methodology presented in this paper unifies previously proposed hybrid NVP-AT schemes and leads to many new variants. Given the extensive literature available in software fault tolerance and continued rapid developments in the field, such unifying methodologies (see, e.g., [49]) are clearly in demand and must be given high priority by the researchers in the field and within our educational programs. A component-based approach is particularly appropriate in that it allows:

- Easy exploration of the vast architectural design space for fault-tolerant software
- Building up of trust and the emergence of trusted components due to reuse
- Swapping of components for more reliable versions as they become available

Continued research in this area will enhance the utility of the proposed general framework for the study of hybrid NVP-AT schemes, leading to more specific design techniques, performance comparisons, and tradeoff guidelines. Results of such extended studies will contribute both to fundamental understanding of voting and acceptance testing as "dependability enhancement" mechanisms [29] and to their practical application in the realization of ultrareliable computations from standard components. A number of specific problems for future investigation are suggested directly by the discussions in the preceding sections of this paper. Examples of promising research directions include:

- Optimal weight augmentation and reduction policies; the $a(w)$ and $r(w)$ functions
- Effects of unequal module complexities and/or reliabilities as well as imperfect voters
- Effects of different voting schemes on optimal configurations and their reliabilities
- Optimal number of modules to be replaced by ATs (parameter k of Fig. 6a)
- Optimal partitioning of n modules for two-level voting (parameter k of Fig. 8)
- More general multilevel voting schemes and their attendant design tradeoffs
- Effects of combined correctness and timeliness requirements [28], [31]

The ultimate goal is to solve the following problem:

> Given a set of components with associated values for p, p', and p'', as well as other system cost and reliability parameters (in particular those characterizing correlated failures), what is the most cost-effective choice and arrangement of computation modules, ATs, and voters?

As this problem is quite challenging, any approach to its solution will necessarily proceed through a number of simpler intermediate problems. For example, one might ask: What is an optimal arrangement of n M/T modules to maximize the overall reliability?

8 References

[1] Ammann P.E. & Knight J.C. (1988) Data Diversity: An Approach to Software Fault Tolerance. *IEEE Trans. Computers*, Vol. 37, pp. 418-425.

[2] Avizienis A. & Kelly J.P.J. (1984) Fault Tolerance by Design Diversity: Concepts and Experiments. *IEEE Computer*, Vol. 17, No. 8, pp. 67-80.

[3] Avizienis A. (1985) The *N*-Version Approach to Fault-Tolerant Software. *IEEE Trans. Software Engineering*, Vol. 11, pp. 1491-1501.

[4] Banerjee P. & Abraham J.A. (1986) Bounds on Algorithm-Based Fault Tolerance in Multiple-Processor Systems. *IEEE Trans. Computers*, Vol. 35, pp. 296-306.

[5] Belli F. & Jedrzejowicz P. (1990) Fault-Tolerant Programs and Their Reliability. *IEEE Trans. Reliability*, Vol. 39, pp. 184-192.

[6] Blum M. & Wasserman H. (1996) Reflections on the Pentium Division Bug. *IEEE Trans. Computers*, Vol. 45, pp. 385-393, 1996.

[7] Butler R.W. & Finelli G.B. (1993) The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software. *IEEE Trans. Software Engineering*, Vol. 19, pp. 3-12.

[8] Chen L. & Avizienis A. (1978) *N*-Version Programming: A Fault Tolerance Approach to Reliability of Software Operation. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 3-9.

[9] Di Giandomenico F. & Strigini L. (1990) Adjudicators for Diverse-Redundant Components. *Proc. Symp. Reliable Distributed Systems*, pp. 114-123.

[10] Dugan J.B. & Lyu M.R. (1994) System-Level Reliability and Sensitivity Analyses for Three Fault-Tolerant System Architectures. *Proc. Int'l Working Conf. on Dependable Computing for Critical Applications*, pp. 295-307.

[11] Dugan J.B. & Lyu M.R. (1994) System Reliability Analysis of an N-Version Programming Application. *IEEE Trans. Reliability*, Vol. 43, pp. 513-519.

[12] Eckhardt D.E. & Lee L.D. (1985) A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors. *IEEE Trans. Software Engineering*, Vol. 11, pp. 1511-1517.

[13] Eckhardt D.E. et al (1991) An Experimental Evaluation of Software Redundancy as a Strategy for Improving Reliability. *IEEE Trans. Software Engineering*, Vol. 17, pp. 692-702.

[14] Gantenbein R.E., Shin S.Y. & Cowles J.R. (1991) Evaluation of Combined Approaches to Distributed Software-Based Fault Tolerance. *Proc. Pacific Rim Symp. Fault-Tolerant Systems*, pp. 70-75.

[15] Gersting, J.L., Nist R.L., Roberts D.B. & Van Valkenburg R.L. (1991) A Comparison of Voting Algorithms for N-Version Programming. *Proc. Hawaii Int'l Conf. System Sciences*, pp. 253-262.

[16] Kelly J., McVittie T. & Yamamoto W. (1991) Implementing Design Diversity to Achieve Fault Tolerance. *IEEE Software*, Vol. 8, pp. 61-71, July.

[17] Kersken M. & Saglietti F., Eds. (1992), *Software Fault Tolerance: Achievement and Assessment Strategies*, Springer.

[18]Kim K.H. & Welch H.O. (1989) Distributed Execution of Recovery Blocks: An Approach to Uniform Treatment of Hardware and Software Faults in Real-Time Applications. *IEEE Trans. Computers*, Vol. 38, pp. 626-636.

[19]Kim K.H. & Kavianpour A. (1993) A Distributed Recovery Block Approach to Fault-Tolerant Execution of Application Tasks on Hypercubes. *IEEE Trans. Parallel & Distributed Systems*, Vol. 4, pp. 104-111.

[20]Kim K.H. (1994) Distributed Execution of Recovery Blocks: An Approach to Uniform Treatment of Hardware and Software Faults. *Proc. Conf. Distributed Computing Systems*, pp. 526-532.

[21]Knight J.C. & Leveson N.G. (1986) An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming. *IEEE Trans. Software Engineering*, Vol. 12, pp. 96-109.

[22]Laprie J.-C., Arlat J., Beounes C., Kanoun K. & Hourtolle C. (1987) Hardware- and Software-Fault-Tolerance: Definition and Analysis of Architectural Solutions. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 116-121.

[23]Laprie J.-C., Arlat J., Beounes C. & Kanoun K. (1990) Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures. *IEEE Computer*, Vol. 23, No. 7, pp. 39-51.

[24]Leveson N.G., Cha S.S., Knight J.C. & Shimeall T.J. (1990) The Use of Self Checks and Voting in Software Error Detection: An Empirical Study. *IEEE Trans. Software Engineering*, Vol. 16, pp. 432-443.

[25]Littlewood B. & Miller D.R. (1989) Conceptual Modeling of Coincident Failures in Multiversion Software. *IEEE Trans. Software Engineering*, Vol. 15, pp. 1596-1614.

[26]Liu C. (1992) A General Framework for Software Fault Tolerance. *Proc. Workshop Fault-Tolerant Parallel & Distributed Systems*, pp. 84-91.

[27]Nicola V.F. & Goyal A. (1990) Modeling of Correlated Failures and Community Error Recovery in Multiversion Software. *IEEE Trans. Software Engineering*, Vol. 16, pp. 350-359.

[28]Parhami B. (1990) A Unified Approach to Correctness and Timeliness Requirements for Ultrareliable Concurrent Systems. *Proc. Int'l Parallel Processing Symp.*, pp. 733-747.

[29]Parhami B. (1991) A Data-Driven Dependability Assurance Scheme with Applications to Data and Design Diversity. in *Dependable Computing for Critical Applications*, Ed. by A. Avizienis and J.C. Laprie, Springer, pp. 257-282.

[30]Parhami B. (1992) Optimal Algorithms for Exact, Inexact, and Approval Voting. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 404-411.

[31]Parhami B. & Hung C.Y. (1993) Scheduling of Replicated Tasks to Meet Correctness Requirements and Deadlines. *Proc. Hawaii Int'l Conf. System Sciences*, pp. 506-515.

[32]Parhami B. (1994) A Multi-Level View of Dependable Computing. *Computers and Electrical Engineering*, Vol. 20, pp. 347-368.

[33]Parhami B. (1994) Voting Algorithms. *IEEE Trans. Reliability*, Vol. 43, pp. 617-629.

[34]Parhami B. (1996) Design of Reliable Software via General Combination of N-Version Programming and Acceptance Testing. *Proc. Int'l Symp. Software Reliability Engineering*, pp. 104-109.

[35]Partridge D. (1997) The Case for Inductive Programming. *IEEE Computer*, Vol. 30, No. 1, pp. 36-41.

[36]Prata P. & Silva J.G. (1999) Algorithm Based Fault Tolerance Versus Result-Checking for Matrix Computations. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 4-11.

[37]Pucci G. (1990) On the Modeling and Testing of Recovery Block Structures. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 356-363.

[38]Randell B. (1975) System Structure for Software Fault Tolerance. *IEEE Trans. Software Engineering*, Vol. 1, pp. 220-232.

[39]Randell B. (1987) Design Fault Tolerance. In *The Evolution of Fault-Tolerant Computing*, Ed. by A. Avizienis, H. Kopetz, and J.-C. Laprie, Springer, pp. 251-270.

[40]Scott K., Gault J.W. & McAllister D.F. (1983) The Consensus Recovery Block. *Proc. Total System Reliability Symp.*, pp. 74-85.

[41]Scott K., Gault J.W. & McAllister D.F. (1987) Fault-Tolerant Software Reliability. *IEEE Trans. Software Engineering*, Vol. 13, pp. 582-592.

[42]Scott R.K. & McAllister D.F. (1996) Cost Modeling of *N*-Version Fault-Tolerant Software Systems for Large *N*. *IEEE Trans. Reliability*, Vol. 45, pp. 297-302.

[43]Siewiorek D.P. & Swarz R.S. (1992) *Reliable Computer Systems: Design and Evaluation*, Digital.

[44]Sitaraman R.K. & Jha N.K. (1993) Optimal Design of Checks for Error Detection and Location in Fault-Tolerant Multiprocessor Systems. *IEEE Trans. Computers*, Vol. 42, pp. 780-793.

[45]Sklaroff J.R. (1976) Redundancy Management Techniques for Space Shuttle Computers. *IBM J. Research & Development*, Vol. 20, pp. 20-28.

[46]Sullivan G.F. & Masson G.M. (1990) Using Certification Trails to Achieve Software Fault Tolerance. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 423-431.

[47]Sullivan G.F. & Masson G.M. (1991) Certification Trails for Data Structures. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 240-247.

[48]Sullivan G.F., Wilson D.S. & Masson G.M. (1995) Certification of Computational Results. *IEEE Trans. Computers*, Vol. 44, pp. 833-847.

[49]Suzuki M., Katayama T. & Schlichting R.D. (1994) Implementing Fault Tolerance with an Attribute and Functional Based Model. *Proc. Int'l Symp. Fault-Tolerant Computing*, pp. 244-253.

[50]Tai A.T., Meyer J.F. & Avizienis A. (1993) Performability Enhancement of Fault-Tolerant Software. *IEEE Trans. Reliability*, Vol. 42, pp. 227-237.

[51]Voges U., Ed. (1988) *Software Diversity in Computerized Control Systems*, Springer.

[52]Xu J. & Randell B. (1997) Software Fault Tolerance: $t/(n-1)$-Variant Programming. *IEEE Trans. Reliability*, Vol. 46, pp. 60-68.