1537

p 28

Computer Science

Technical Report

Software Reliability Through Fault-Avoidance and Fault-Tolerance

Report #4 (9/1/90-2/28/91) on NAG-1-983

by

Mladen A. Vouk and David F. McAllister

North Carolina State University

Box 8206
Raleigh, NC 27695

(NASA-CR-187998) SOFTWARE RELIABILITY THROUGH FAULT-AVOIDANCE AND FAULT-TOLERANCE Semiannual Technical Report No. 4, 1 Sep. 1990 - 28 Feb. 1991 (North Carolina State Univ.) 28 p CSCL 09B G3/61

N91-19741

Unclas 0001537

Semi-Annual Technical Report Submitted to the

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Langley Research Center, Hampton, Va.

for research entitled

SOFTWARE RELIABILITY THROUGH FAULT-AVOIDANCE AND FAULT-TOLERANCE

(Report #4 on grant NAG-1-983)

from

Mladen A. Vouk, Co-Principal Investigator, Assistant Professor
David F. McAllister, Co-Principal Investigator, Professor

Department of Computer Science
North Carolina State University
Raleigh, N.C. 27695-8206
(919) 737-2858

Report Period
Beginning Date: September 1, 1990.
Ending Date: February 28, 1991.

An Empirical Evaluation of Some Software Fault-Tolerance Schemes in the Presence of Failure Correlation*

Mladen A. Vouk, David F. McAllister, Kalhee Kim

Department of Computer Science
North Carolina State University, Box 8206
Raleigh, NC 27695-8206

Tel: (919) 737-7886
Fax: (919) 737-7382
e-mail: vouk@adm.csc.ncsu.edu

Key Words:
Consensus Voting, Consensus Recovery Block, Acceptance Voting, System Reliability, Software Fault-Tolerance

Abstract

We used 20 independently developed but functionally equivalent software versions to investigate and compare empirically some properties of N-version programming, Recovery Block and Consensus Recovery Block using the majority and consensus voting algorithms. We have also compared this with another hybrid fault-tolerant scheme called Acceptance Voting using dynamic versions of consensus and majority voting.

Consensus voting provides adaptation of the voting strategy to varying component reliability, failure correlation, and output space characteristics. Since failure correlation among versions effectively reduces the cardinality of the space in which voters make decisions, consensus voting is usually preferable to simple majority voting in any fault-tolerant system.

When versions have considerably different reliabilities, the version with the best reliability will perform better than any of the fault-tolerant techniques. Consensus Recovery Block produces the highest reliability most of the time. It outperforms N-version programming, but also very successfully competes with Recovery Block in situations where the acceptance test is not of the highest quality. Consensus Recovery Block is surprisingly robust even in the presence of failure correlation.. Acceptance voting is, under special circumstances, more reliable than both Consensus Recovery Block and Recovery Block. It, in general, has lower reliability than the others.

*Research supported in part by NASA Grant No. NAG-1-983

1. Introduction

1.1 N-Version Programming and Recovery Block

In many fault-tolerant software systems, redundancy is used to provide the fault-tolerance. Several independently developed but functionally equivalent (IDFE) software versions are combined in various ways in an attempt to increase reliability and safety .

One of the earliest fault-tolerant systems (FTS) was **Recovery Block (RB)** [e.g., Ran75] in which IDFE versions are executed in sequence and the output passed to an acceptance test (AT). If the output of the first version fails the acceptance test, then the second or backup software version is executed and its output checked by the AT, etc. In the case that the outputs of all versions are rejected the system fails. A problem with this strategy is the sequential nature of the execution of versions and finding a simple and highly reliable acceptance test which does not involve the development of an additional software version. As a result, other FTS strategies such as **N-version Programming (NVP)** [e.g., Avi77, Avi85] were proposed where all IDFE versions are executed in parallel and their outputs adjudicated by a voter. A problem with strategies based on voting is that situations can arise where there is an insufficient number of agreeing versions and voting fails simply because the voter cannot make a decision. In the following sections we will investigate empirically several alternative voting strategies for these and other fault-tolerant schemes. Two hybrid fault-tolerant software methods which combine RB and NVP are examined.

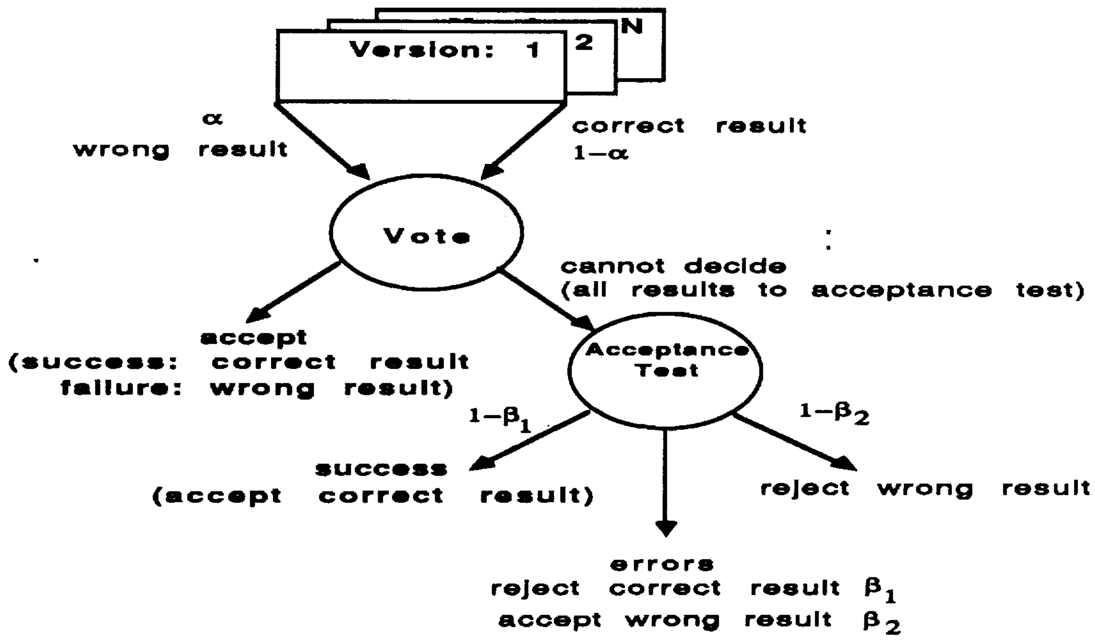

Figure 1. The Consensus Recovery Block model.

1.2 Consensus Recovery Block

Scot et al. [Sco87] have developed a hybrid software fault-tolerance model called Consensus Recovery Block (CRB). The system first attempts to vote. If the voting module cannot make a decision, the system reverts to RB. The strategy is depicted in Figure 1, where the number of versions in the system is N, $1 - \alpha$ is the probability that a version gives correct result, β_1 is the probability that acceptance test rejects a correct result, β_2 is the probability that acceptance test accepts an incorrect result. In general, CRB offers system reliability superior to that provided by NVP[Sco87, Bel90]. However, CRB, like NVP does not resolve the problem of a voter which returns a wrong answer because several versions produce identical-and-wrong answers or there is not a majority as might be the case when there are multiple correct outputs.

1.3 Acceptance Voting

A hybrid fault-tolerant method we call Acceptance Voting (AV) was first described in [Ath89] The method is also described in [Bel90]. Athavale [Ath89] developed the model specifically as a solution in some situations where CRB may have problems, e.g. small output spaces. [Bel90] proposes the method as a possible solution for poor acceptance tests. We are more interested in the former since the performance of some more reliable hybrid models such as CRB deteriorates when the output space is small. A binary output space is an extreme case where CRB acts as a simple voter, and the acceptance test is never invoked. One way of treating this problem is to reduce, or completely eliminate, as many wrong answers as possible before voting. This, however, reduces the voter decision space.

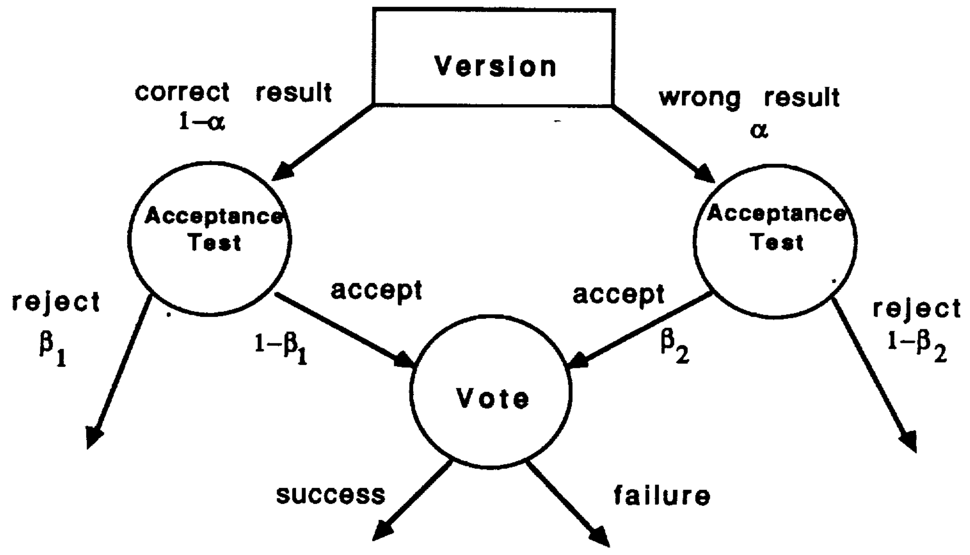

Figure 2. Block diagram of Acceptance Voting

The model of the AV scheme is shown in Figure 2. As in NVP and RB, N IDFE versions are developed, together with an acceptance test, and a voting procedure. When AV is invoked, all versions execute and submit their outputs for acceptance testing. Only the outputs that pass the acceptance test continue on to the voter. Each time the model is invoked it may vote with a different number of outputs, depending on how many results were passed to the voter by the AT. We will investigate various voting schemes for handling the case when some of the versions have failed the acceptance test.

2. Adjudication Strategies

2.1 Majority and 2-out-of-N Voting

In an m-out-of-N fault-tolerant software system the number of versions is N, and m is the agreement number, or the number of matching outputs which the voting or adjudication algorithm requires for system success [e.g. Tri82, Eck85]. In the past, because of cost restrictions, N was rarely larger than 3, and m was traditionally chosen as $\frac{N+1}{2}$ for odd m. In general, in **Majority Voting (MV)**, $m = \lceil \frac{N+1}{2} \rceil$, where $\lceil\ \rceil$ denotes the ceiling function. Scott et al. [Sco87] showed that, if the output space is large, and true statistical independence of version failures can be assumed, there is no need to choose $m > 2$ regardless of the size of N, although larger m values offer additional benefits. We will use the term **2-out-of-N Voting (2NV)** for the case where agreement number m=2. In this experiment we do not have statistical independence of version failures. Hence, we will only use this voting technique when showing upperbounds for reliabilities of the systems we consider. In a model based on software diversity and a voting strategy there is a difference between correctness and agreement. McAllister et al. [McA90] distinguish between agreement and correctness and develop and evaluate an adaptive voting strategy called **Consensus Voting (CV)**. This strategy is particularly effective in small output spaces because it automatically adjusts the voting to the changes in the effective output space cardinality. They show that the MV strategy provides a lower bound on the reliability provided by CV, and that a 2-out-of-N voting strategy gives an upper bound for $m \geq 2$.

2.2 Consensus Voting

In Consensus Voting the voter uses the following algorithm to select the "correct" answer:

- If there is a majority agreement ($m \geq \lceil \frac{N+1}{2} \rceil$, N>1) then this answer is chosen as the " correct" answer.

- Otherwise, if there is a unique maximum agreement, but this number of agreeing versions is less than $\lceil \frac{N+1}{2} \rceil$, then this answer is chosen as the "correct" one.

- Otherwise, if there is a tie in the maximum agreement number from several output groups then
  - if CV is used in NVP one group is chosen at random and the answer associated with this group is chosen as the "correct" one.
  - else if CV is is used in CRB all groups are subjected to an acceptance test which is then used to choose the "correct" output.

In [McA90] it is shown that the strategy is equivalent to MV when the output space cardinality is 2, and to the 2-out-of-N voting when the output space cardinality tends to infinity provided the agreement number is not less than 2. It is also proved that, in general, the boundary probability below which the system reliability begins to deteriorate as more versions are added is $\frac{1}{r}$, where r is the cardinality of the output space.

2.3 Dynamic Voting

In the AV fault-tolerant scheme the number of outputs passed to the voter may vary. Voting may be done using any suitable voting scheme. The voting agreement number may be chosen statically (e.g. 2 of N, or majority of N) but the situation may arise where there are less outputs than the agreement number in which case the voter cannot make a decision. A better technique is to choose a voting algorithm which is dynamic and depends on the number of outputs passed to the voter. We call the latter approach dynamic voting and distinguish between Dynamic Majority Voting (DMV) and Dynamic Consensus Voting (DCV) .The difference between the DMV and MV is that even if a small number of results are passed to the voter, dynamic voting will try to find the majority among them. Both Athavale [Ath89] and Belli and Jedrzejowicz [Bel90] have limited their analytical work to DMV. In the following sections we also explore AV with DCV.

2.4 Coincident Failures

Most of the theory developed in [Sco87, McA90, Ath89, Bel90] was derived under the assumption of interversion failure independence. When two or more functionally equivalent software components fail on the *same* input case we say that a *coincident* failure has occurred. When two or more versions give the same incorrect response, to a given tolerance, we say that an *identical-and-wrong* (IAW) answer was obtained. If the measured probability of the coincident failures is significantly different from what would be expected by random chance (using failure independence model) then we say that the observed coincident failures are *correlated* or *dependent*. Experiments have shown that inter-version failure dependence among IDFE versions may not be negligible in the context of current software development and testing strategies [e.g., Sco84a, Kni86]. Reliability performance of NVP in the presence of failure correlation was theoretically investigated in [Sco84a, Sco84b, Eck85, Lit90].

In this paper we empirically examine the behavior of CRB, AV, NVP and RB models in the presence of failure correlation. For this we use 20 IDFE programs developed in a multiversion experiment [Kel88, Vou90]. The primary goal of this study is investigation of the properties of CV, CRB and AV, and not of the principle of software diversity and the faults that may be associated with it. The versions are used only as a medium for testing the hypotheses about the above models. We will argue that if choosing a wrong answer or having no answer has the same impact on the system, then these models should be preferred to strategies like majority voting even in the presence of correlated failures.

Table 1. Version failure rates.

Version	Failure Rate*	
	Estimate I	Estimate II
1	0.58	0.59
2	0.07	0.07
3	0.13	0.11
4	0.07	0.06
5	0.11	0.10
6	0.63	0.64
7	0.07	0.06
8	0.35	0.36
9	0.40	0.39
10	0.004	0.000
11	0.09	0.10
12	0.58	0.59
13	0.12	0.12
14	0.37	0.38
15	0.58	0.59
16	0.58	0.59
17	0.10	0.09
18	0.004	0.006
19	0.58	0.59
20	0.34	0.33

(*) Based on the 3 "best.acceleration" variables. Each column was obtained on the basis of a separate set of 500 random cases.

3. Empirical Results

3.1 Experimental Environment

The 20 software versions we used are described in [Kel88] and [Vou90]. The versions and execution profiles were known to result in relatively high intensity correlated program failures. In fact, we used the program versions before they underwent the so called "certification testing" phase of the above experiment in order to retain the mix of faults present immediately after unit development phase. To investigate empirically the properties of different FTS models we needed subsets of versions with properties such as: 1) similar average[1] N-tuple reliability (where N is the number of versions assumed to be involved in decision making), 2) a range of average N-tuple reliabilities, and 3) a range of voter decision space cardinalities. In this paper we report on the behavior of 3, 5 and 7 version systems formed from the pool of 20 versions. The subset selection process is described in Appendix I.

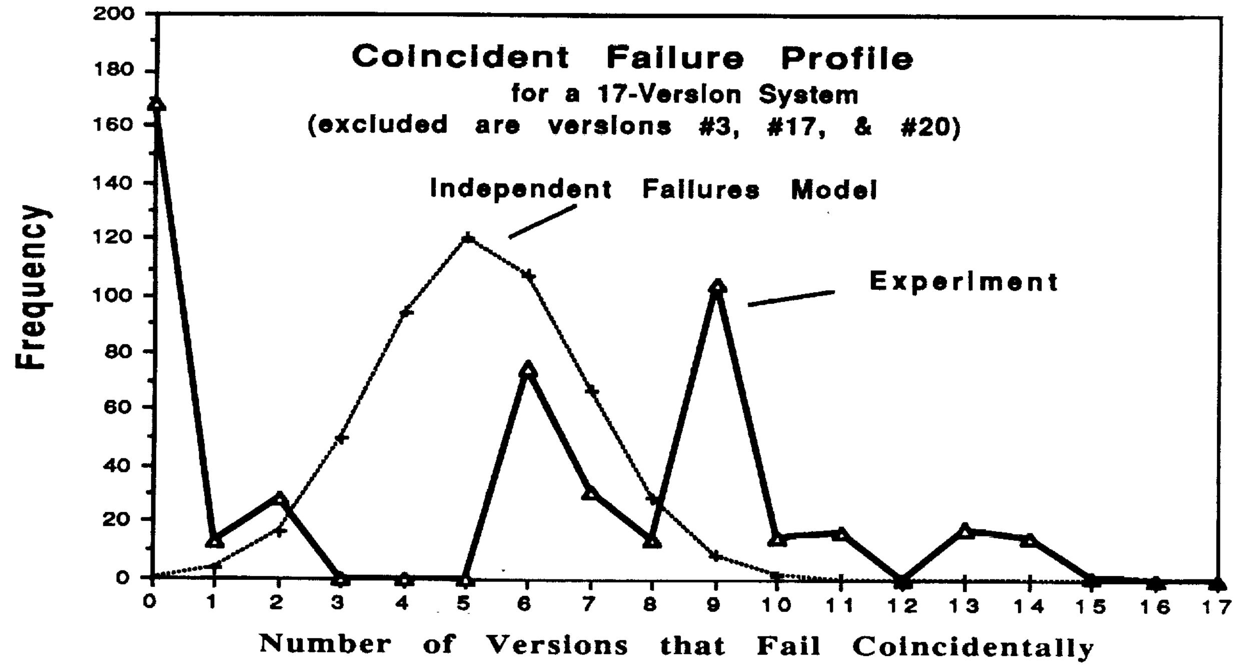

Figure 3. Example of a joint coincident failure profile.

The test set profile, the choice of output variables involved in the voting, and the number and the choice of versions all influence the mix and intensity of observed failures. In conducting our experiments we considered a number of input profiles and different combinations of versions and output variables. Failure rate estimates based on the three most critical output variables (out of 63 monitored) are shown in Table 1. Two test suites each containing 500 uniform random input test cases were used in all estimates discussed in this paper. One suite was used to obtain estimates of individual version failure rates, N-tuple reliabilities, select sample N-tuple combinations, and compute expected "independent model" response. The other was used to investigate the behavior of the voting and fault-tolerance strategies. The sample size is sufficient for the version and N-tuple

[1] Estimated average N-tuple reliability = $\bar{p} = \sum_{i=1}^{N} \frac{\hat{p}_i}{N}$, sample stand. dev. estimate = $\hat{\sigma} = \sqrt{\sum_{i=1}^{N} \frac{(\bar{p}-\hat{p}_i)^2}{N-1}}$, where $\hat{p}_i$ is estimated reliability of version i.

reliability ranges on which we report here[2]. Acceptance test used in RB, AV and CRB studies was one of the 20 versions not in the subset from which N-tuples were drawn. This provided possible correlation not only among versions, but also between the acceptance test and the versions. The FTS algorithms of interest were invoked for each test case. The outcome was compared to the correct answer obtained from a "golden" program [Kel88, Vou90] and the frequency of successes and failures for each strategy was recorded.

The failure correlation properties of the version sets can be deduced from their joint coincident failure profiles, and the corresponding identical-and-wrong response profiles. Figure 3 shows the profile for a 17 version system (three versions selected to act as acceptance tests are not in the set). The abscissa represents the number of versions that fail coincidentally, and the ordinate is the frequency of the event over the 500 test suite samples. Also shown is the expected frequency profile corresponding to the independent "binomial" model [Tri82, Vou85]. The deviation from the "expected" profile is obvious. Table 2 summarizes the corresponding empirical frequency of coincident identical-and-wrong responses. For example, in 500 tries there were 15 events where 8 versions coincidentally returned an answer which was wrong yet identical within the tolerance used to compare the three most critical (real) variables. Both, Figure 3 and Table 2 are strong indicators of a high degree of inter-version failure dependence.

Table 2. Frequency of empirical coincident identical-and-wrong (IAW) events over 500 test cases for the set of 17 versions shown in Figure 3. The span is the number of versions that coincidentally returned a IAW answer.

The Span of IAW Events																
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Frequency																
2049	164	1	16	1	1	2	15	0	0	0	0	0	0	0	0	0

We study how different FTS models behave in this environment. We investigated NVP with MV (NVP-MV) , and NVP with CV (NVP-CV). We also considered RB, CRB with MV (CRB-MV), CRB with CV (CRB-CV), and AV. With Acceptance Voting we used Dynamic Majority Voting (AV-DMV) and Dynamic Consensus Voting (AV-DCV).

3.2 Consensus Voting

[2]A investigation using higher reliability versions and up to 100,000 test cases is in progress.

Table 3 gives examples of the detailed behavior of selected individual N-tuples. In the table we show the count of the number of times the best version in an N-tuple was correct (Best Version), and the success frequency of each of the above fault-tolerant strategies. The best response is underlined with a full line, while the second best with a wavy line. Also shown in the table is the breakdown of the NVP-CV decision process by the frequency of sub-events that yielded the consensus decision. We recorded the number of times consensus was a successful majority (S-Majority), an unsuccessful majority (F-Majority), a successful plurality (S-Plurality) which corresponds to the situation where there is a unique maximum of identical outputs but that maximum is not a majority, an unsuccessful plurality (F-Plurality), a successful (S-Random) and an unsuccessful (F-Random) attempt at breaking a tie by random selection, and a failure by fiat (F-Fiat) by which we mean a situation where a tie existed but all the groups of outputs involved contained wrong answers so any choice made to break the tie led to failure. The sum of S-Majority, S-Plurality and S-Random comprises consensus voting success total, while the sum of F-Majority, F-Plurality, F-Random and F-Fiat is equal to the total number of cases where voting failed (F-Total).

The maximum voter decision space for a single test case is N. The table shows the average conditional voter decision space (CD-space) which is defined as the average size of the space in which the voter makes the decisions given that at least one of the versions has failed. We use CD-space to focus on the behavior of the voters when failures are present.

Figures 4 and 5 illustrate the relationship between NVP-CV and NVP-MV success over a range of average N-tuple reliabilities for 3-version and 7-version systems respectively. The "ragged" look of the experimental traces is partly due to the small sample (500 test cases), but also due to the presence of very highly correlated failures. The experimental behavior is in good agreement with the trends indicated by the theoretical CV model based on failure independence [McA90]. For instance, we see that for N=3, NVP has difficulty competing with the best version when the average N-tuple reliability is low. This is because the selected low average reliability N-tuples are composed of versions which are not "balanced", i.e. their reliabilities are very different. As average N-tuple reliability increases (and N-tuples become more balanced) NVP performance approaches, or exceeds, that of the best version. We also see that larger N improves performance of CV more than it does that of MV.

The columns 1 and 2 of Table 3 show the results for two unbalanced low reliability 3-tuples, while column 3 shows the results for a well balanced 3-tuple of higher reliability. We see that in the former case the highest reliability is that of the best version while in the latter NVP-CV offers the best result.

An examination of CV sub-events shows that in the case of low reliability 3-tuples most of the voting success counts come from majority agreement. The rest of the cases result in the failure of the NVP-MV model because all three versions return different results. However, CV attempts to salvage the situation. For instance, for 3-tuple in column 1 CV attempted to recover 289 times by random selection of one of the outputs, and as would be expected it succeeded about 30% of the time. In two cases all three answers returned by the 3-tuple were wrong and failures occurred regardless of which strategy was employed (F-Fiat).

Table 3. Examples of the frequency of voting and recovery events.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
N-tuple Structure																	
Versions	6,10,16	8,9,18	7,11,13	2,3,5 7,11, 14,20	2,3,4 9,11, 13,20	1,3,8, 9,11, 15,16	5,6,8 10,12, 16,20	4,7,8, 12,15	6,9,12 13,14	4,7,8 11,13	1,3,5 8,20	5,7,11 13,20	8,9,18	8,9,18	3,5,9 11,20	3,5,10 11,17	4,5,8 12,20
Mean Value																	
Avg. Rel.	0.59	0.75	0.91	0.91	0.79	0.71	0.60	0.63	0.58	0.86	0.70	0.86	0.75	0.75	0.79	0.91	0.71
Std. Dev.	0.35	0.21	0.03	0.03	0.15	0.18	0.26	0.23	0.20	0.12	0.19	0.11	0.21	0.21	0.14	0.05	0.21
ATRel.	0.87	0.90	0.87	0.996	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.87	0.996	0.996	0.90
CD-Space	2.92	2.57	2.13	2.50	2.98	3.26	5.28	4.19	4.21	2.43	2.82	2.42	2.56	2.56	2.47	2.46	2.65
Std. Dev.	0.27	0.50	0.34	0.79	1.03	1.26	1.17	1.38	0.91	0.73	0.87	0.66	0.50	0.50	0.85	0.78	0.77
Success Frequency																	
Best Version	498	498	467	467	466	454	498	467	441	467	447	467	498	498	454	498	466
NVP-MV	209	344	484	464	436	405	209	285	287	466	400	464	344	344	450	464	432
NVP-CV	302	384	489	483	475	469	449	475	348	482	465	491	384	381	472	488	454
RB	432	456	434	486	455	455	449	454	450	450	436	459	456	433	474	492	453
CRB-MV	435	465	484	483	471	471	449	469	465	466	449	492	465	419	475	493	465
CRB-CV	435	465	484	483	467	483	464	467	465	482	466	493	465	419	476	493	465
AV-MV	432	449	434	482	450	450	450	450	450	450	434	450	449	433	481	482	450
AV-CV	432	458	434	482	450	451	450	450	450	450	434	458	458	433	481	490	450
Success Frequency of CV Sub-Events																	
S-Majority	209	344	484	464	436	405	209	285	287	466	400	464	344	344	450	464	432
F-Majority	0	34	0	15	1	0	0	0	0	0	15	0	34	34	0	0	18
S-Plurality	0	0	0	19	31	62	238	181	42	16	65	20	0	0	15	19	17
F-Plurality	0	0	0	1	17	15	33	18	1	16	2	0	0	0	0	0	1
S-Random	93	40	5	0	8	2	2	9	19	0	0	7	40	37	7	5	5
F-Random	196	82	10	1	7	16	18	7	120	2	2	8	82	85	11	12	12
F-Fiat	2	0	1	0	0	0	0	0	31	0	16	1	0	0	17	0	15
F-Total	198	116	11	17	25	31	51	25	152	18	35	9	116	119	28	12	46

(*) Assuming that effective error output space has infinite cardinality, i.e. there are no coincident identical and wrong answers from two or more versions.

Columns 4-7 illustrate behavior of 7-tuples, and columns 8-12, and 15-17 behavior of 5-tuples. When N > 3 advantages of CV over MV increase because plurality vote is now possible. One problem that NVP-MV does not solve are the small space situations where the vote fails because a

ORIGINAL PAGE IS
OF POOR QUALITY

voter is offered more than two groups of answers to select the "correct" output from, but there is no majority so voting cannot return a decision. The events are those where there is no agreement majority but one of the outputs occurs more frequently than any other, and those where there is a tie between the maximum number of outputs in two or more groups of outputs. For example, consider the 5-version system from column 8 where CV is the best overall strategy. Correct majority was available in only 285 cases, while in 181 instances the correct output was chosen by plurality, and 9 times success came by random selection.

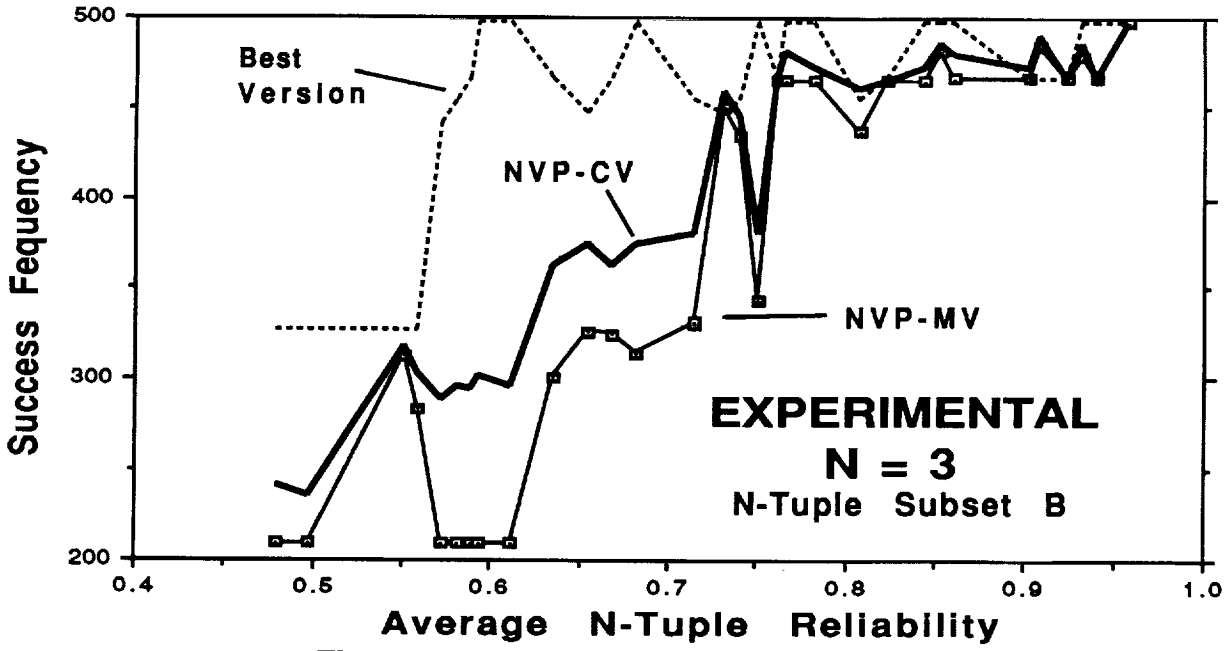

Figure 4. System reliability by voting (N=3).

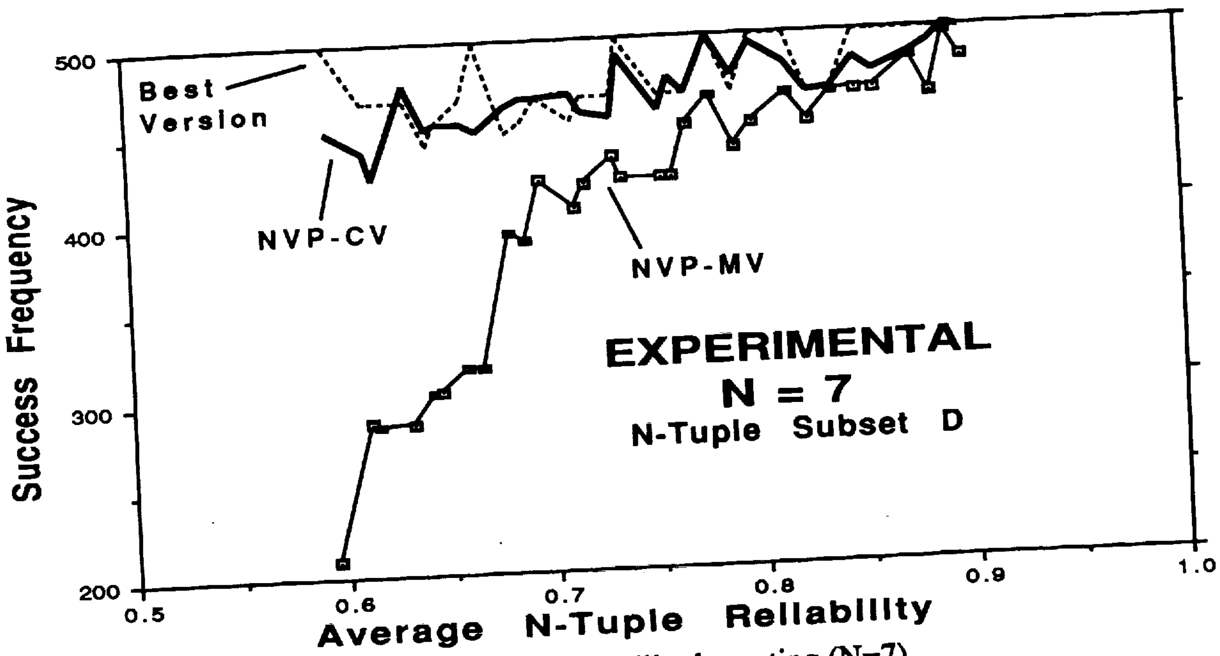

Figure 5. System reliability by voting (N=7).

The theoretical relationship between the system output space cardinality, r, and voting strategies is illustrated in Figure 6 for N=15. In the figure we plot the reliability of NVP-CV versus the average reliability of a 15-version system for different output space cardinality values. Also shown are the NVP 2-out-of-N and NVP-MV boundary curves. It is important to note that both MV and 2-out-of-N voting are effectively output space insensitive and that 2-out-of-N is a viable strategy only when r >> 1. CV is r sensitive and therefore will perform better than MV for $r > 2$. The reliability based on MV is a lower limit on the reliability of CV, while 2-out-of-N voting reliability is an upper limit on the reliability by CV when agreement number is $m \geq 2$.

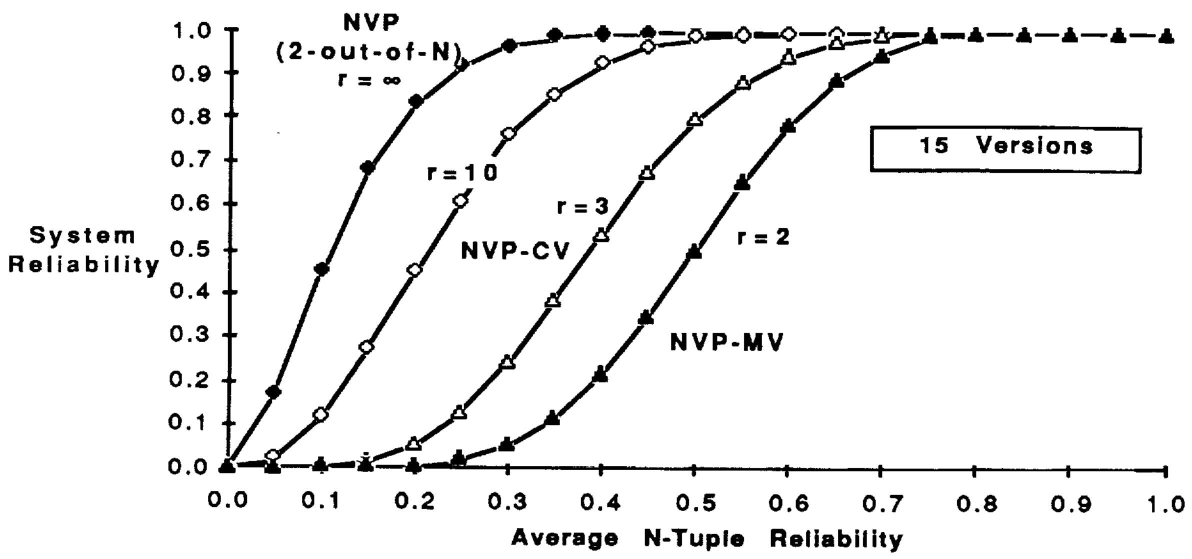

Figure 6. System reliability under different voting strategies. The probability of each j=2,..,r failure state is $\frac{1-p}{r-1}$. All versions in the 15-version system are assumed to have same reliability p.

Figure 7 illustrates the influence of the the average conditional voter decision space for a subset of 5-version systems that have an approximately equal reliability. Again the behavior is in good agreement with the theoretical trends shown in Figure 6. The larger the decision space cardinality the more reliable NVP-CV becomes over NVP-MV. In Figure 7 the maximum decision space is 5. The other extreme is binary output space (r=2). An answer is either correct, or incorrect and no distinction is possible among incorrect answers. In binary output space CV reduces to NVP-MV and cannot improve on it.

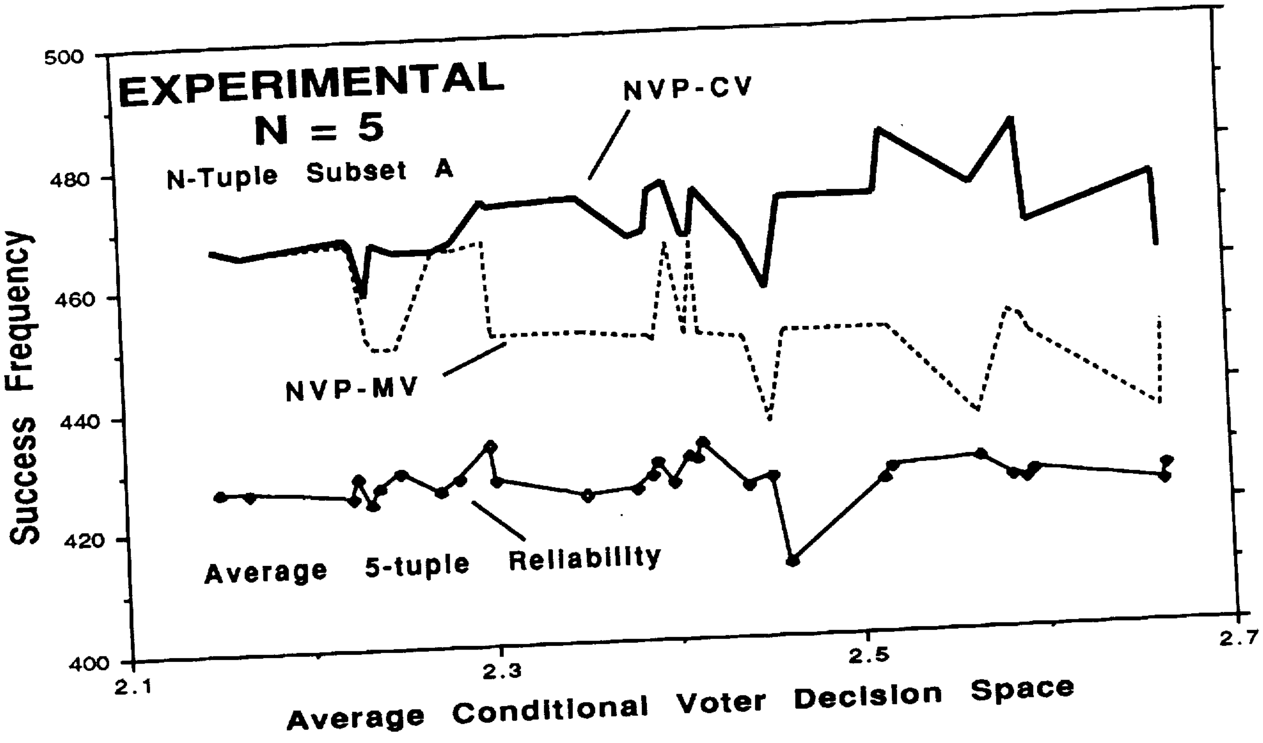

Figure 7. Voter behavior in small decision space.

In practice, failure probabilities of individual versions have a nonzero standard deviation about the N-tuple mean. Small scatter may, up to a point, increase average reliability obtained by voting because there may be enough versions on the "high" side of the mean to form a correct agreement number more often than would be expected from a set where all versions have the same reliability. But when the scatter is excessive the system reliability can actually be lower than the reliability of one or more of its best component versions. This effect is illustrated in Figure 8 [McA90]. The data were obtained by simulation based on the independent model (100,000 case runs for each point shown). We plot the system reliability based on NVP-CV and NVP-MV against the standard deviation of the N-tuple reliability (the mean value being constant). Also shown is the reliability of the best single version involved in the simulation. The feature to note is the very sharp step in the best version reliability once some critical value of the standard deviation of the sample is exceeded (about 0.03 in this example). The same effect was observed experimentally. It can be seen in Figures 4 and 5, and Table 3 examples, where low average reliability systems with a high standard deviation about the average perform worse than the best version. It is better illustrated in Figure 9 where we see the reliability of 5-version systems with average reliability of about 0.85 vs. the standard deviation of the 5 version sample.

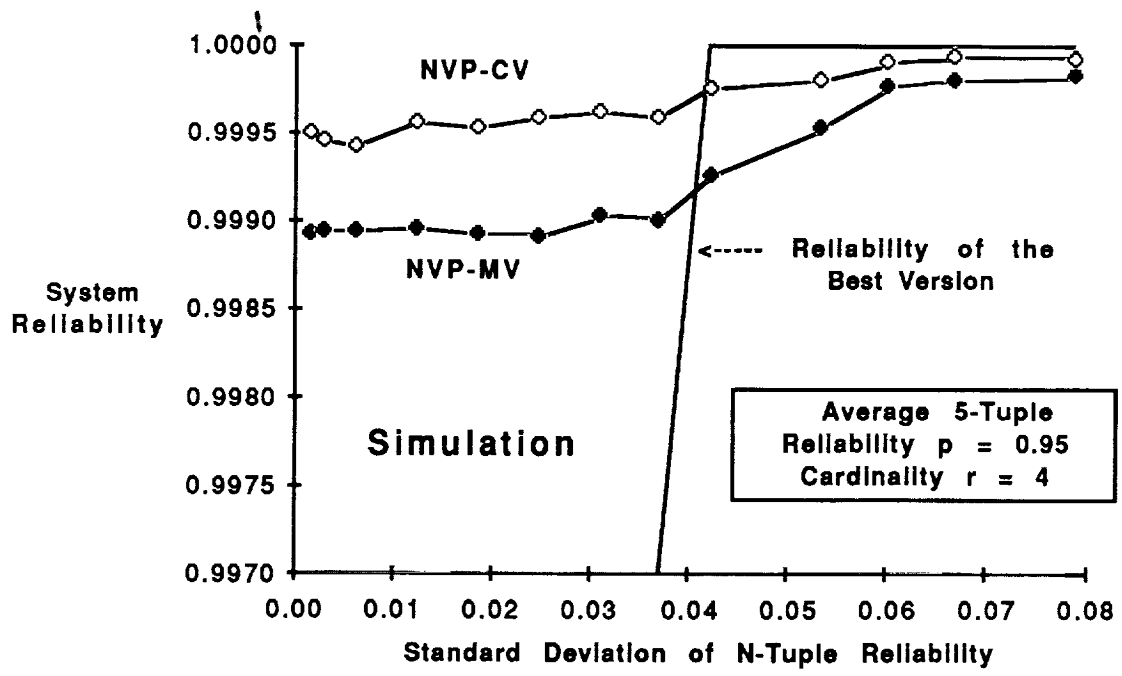

Figure 8. System reliability by CV for 5-version systems vs. standard deviation of 5-tuple reliability. The probability of each j=2,..,r failure state is $\frac{1-p}{r-1}$, where p is the average 5-tuple reliability.

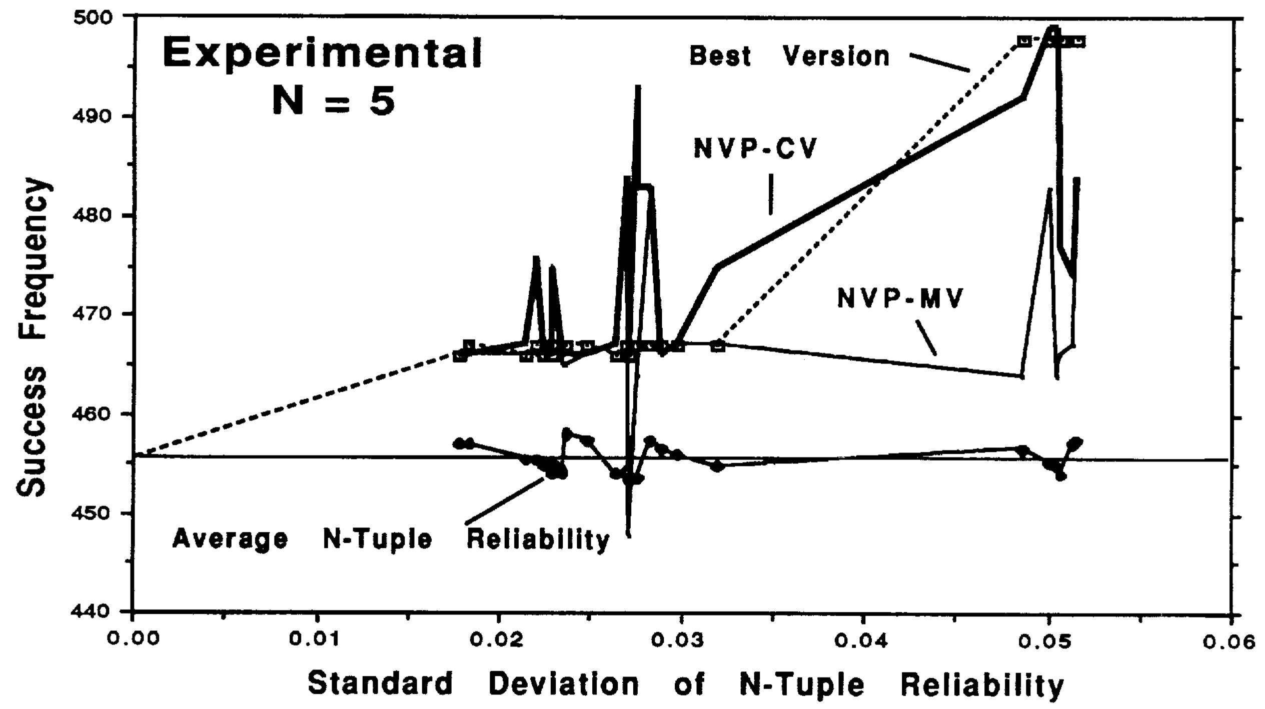

Figure 9. Influence of small voter decision space

From Figure 9 we see that NVP-CV is relatively successful in providing reliability comparable to, or better than, that of the best version so long as the dispersion around the mean is relatively small. The very high variability in the performance of both NVP strategies is due to highly correlated failures. In general, presence of faults resulting in correlated failures will produce an effect which is equivalent to either a reduction, or an increase in the average output space cardinality. In situations like that a voting strategy which can adapt to changes in the effective output space cardinality may have definite advantages over NVP-MV (which fails if there is no majority).

Inspection of Table 3 and Figures 4, 5, 7 and 9 confirms that NVP-CV behaves very much like its model based on failure independence in small output spaces predicts. For example, CV always offers reliability at least equivalent to MV, and performs better than MV when the average decision space in which voters work is sufficiently large.

3.3 Recovery Block and Consensus Recovery Block

Theory predicts that in an ideal situation (version failure independence, most favorable voting strategy, perfect voter) CRB is always superior to both NVP (given the same version reliabilities and the same voting strategy) and Recovery Block (given the same version and acceptance test reliabilities) [Sco87]. This is illustrated in Figure 10. It is interesting to note the cross-over point between RB and NVP caused by low reliability of the RB acceptance test.

The experimental results are illustrated in Figures 11 and 12. The number of times that the result provided by the FTS strategy was correct is plotted against the version reliability. The same acceptance test version was used by CRB and RB. From the figures we see that CRB-MV provides reliability always equal to or larger than the reliability by NVP-MV (given the same versions), and reliability which is usually at least as good or better than that by RB. Usually CRB-CV is better than NVP-CV, but because CRB-CV employs the acceptance test to resolve situations where there is no plurality while NVP-CV uses random tie breaking, occasionally NVP-CV is marginally more reliable than CRB-CV. Examples of this can be seen in columns 3, 5, and 8 or Table 3, and Figures 12 and 17. In all three tabulated cases the difference in favor of NVP-CV is exactly equal to S-Random.

The larger the number of versions (N-tuple) the better the performance of CRB when compared to NVP (given the same voting strategy) or RB (given the same acceptance test). We observe that NVP-CV and NVP-MV may not perform as well as RB given a sufficiently reliable acceptance test, but CRB-CV (or CRB-MV) is generally superior (Figures 11 and 12).

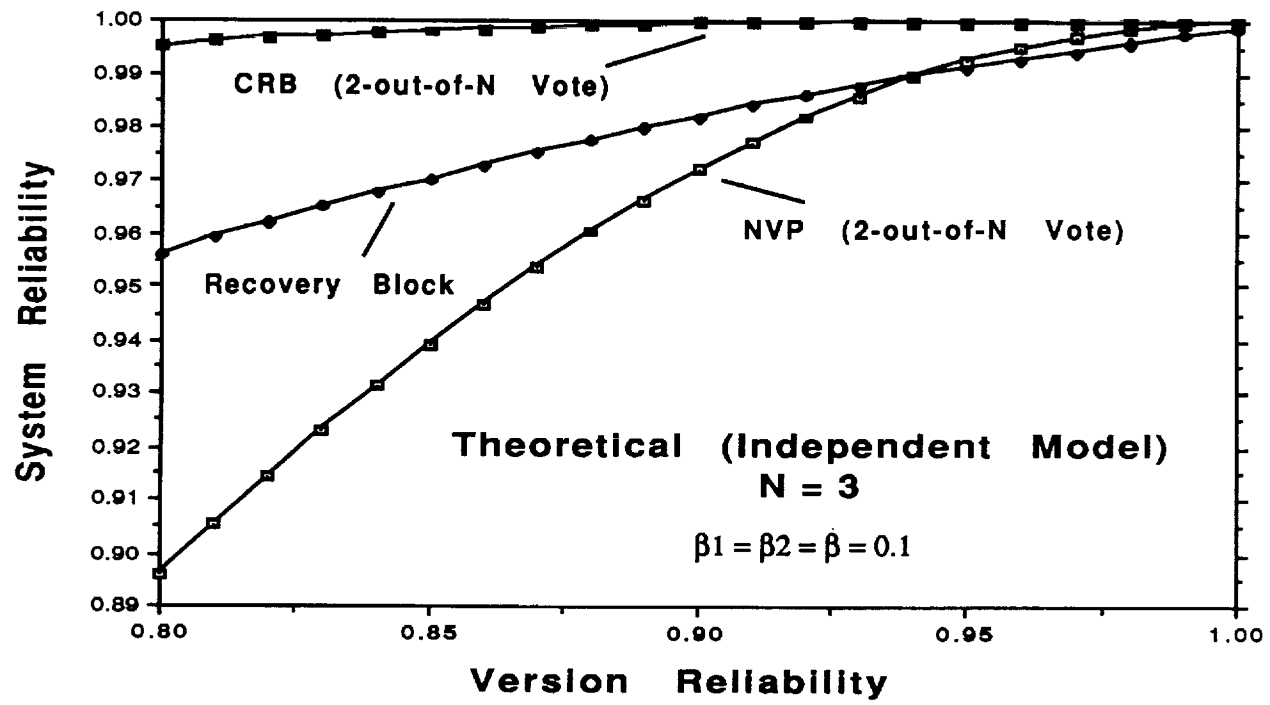

Figure 10. System reliability for different software fault tolerance schemes with 2-out-of-N voting, N = 3, and $\beta_1 = \beta_2 = \beta = 0.1$ (see Figure 1).

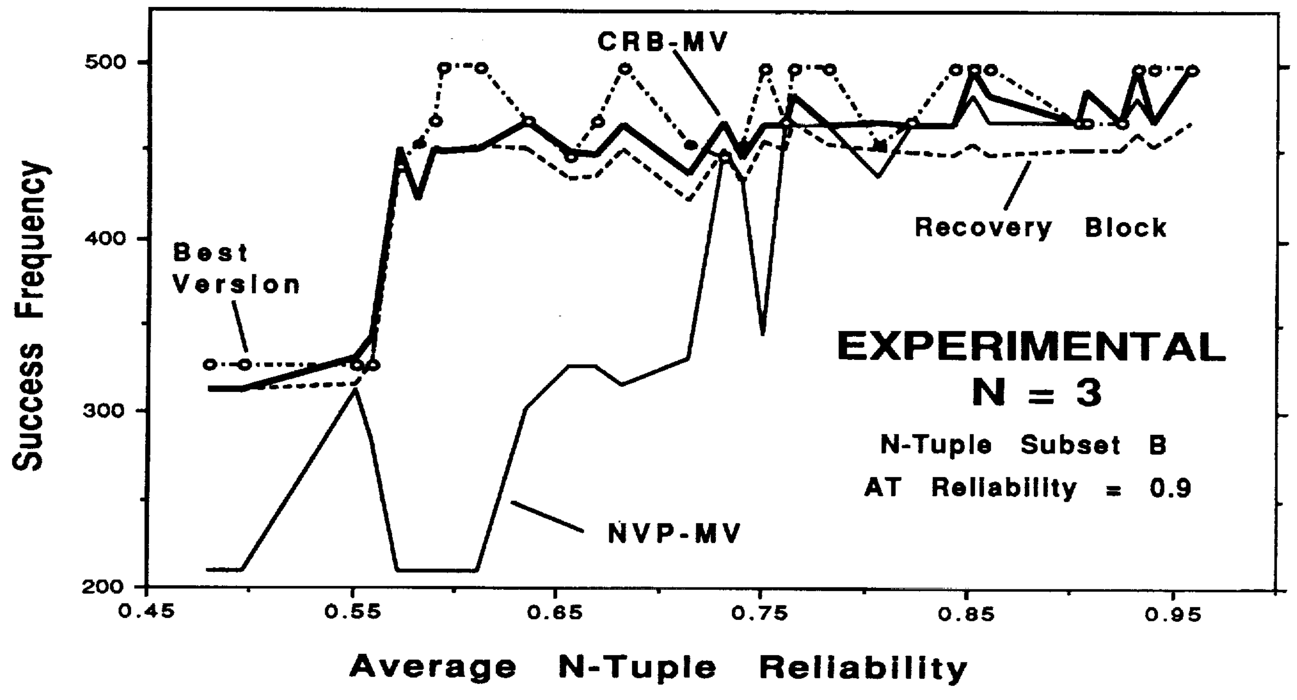

Figure 11. CRB system reliability with majority voting.

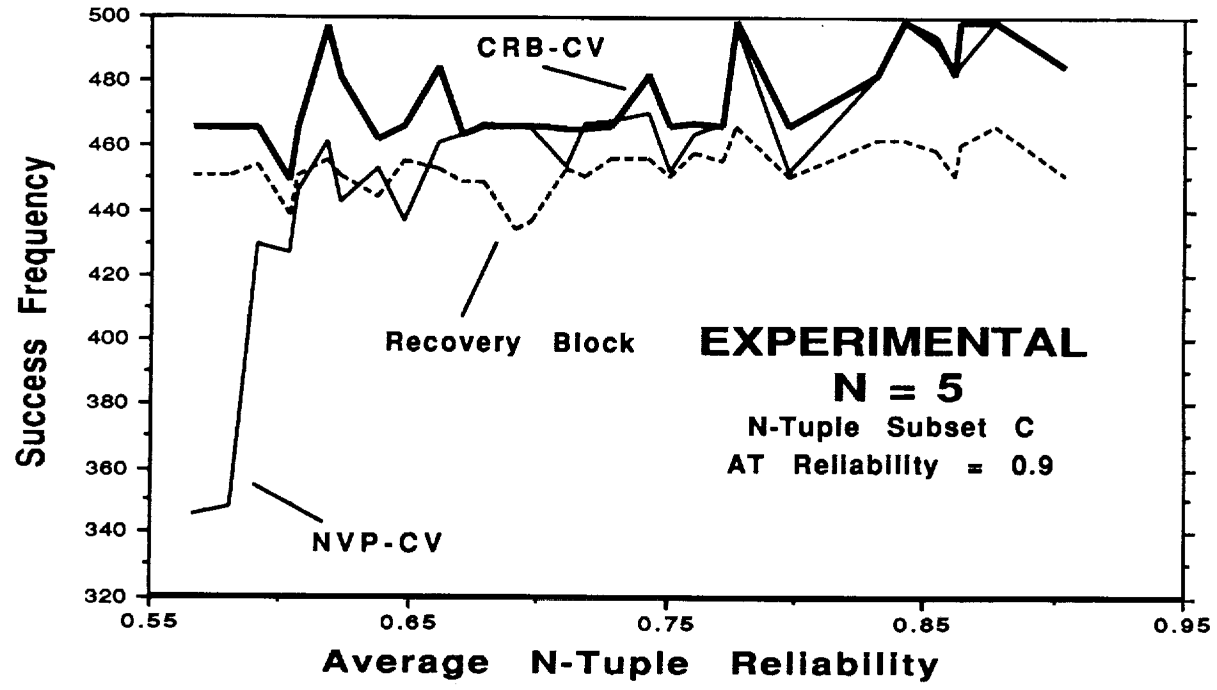

Figure 12a. CRB-CV compared with NVP-CV and RB.

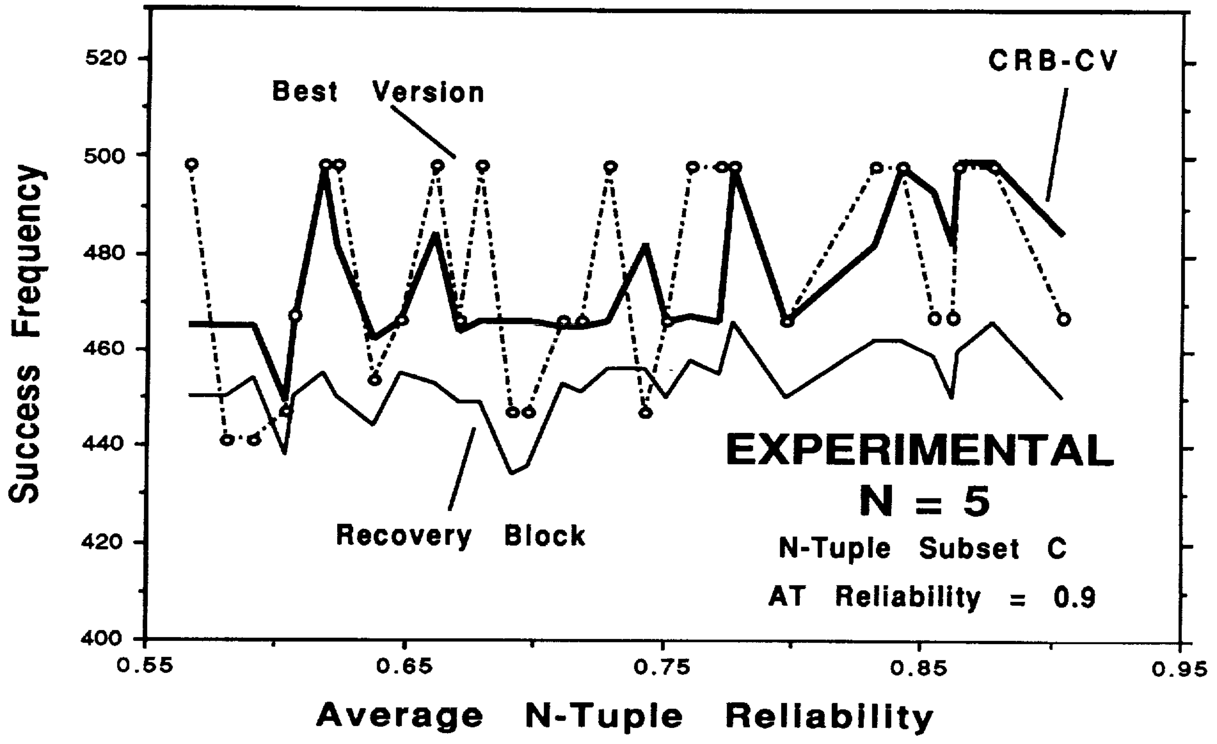

Figure 12b. Comparison of CRB-CV with RB and best version success.

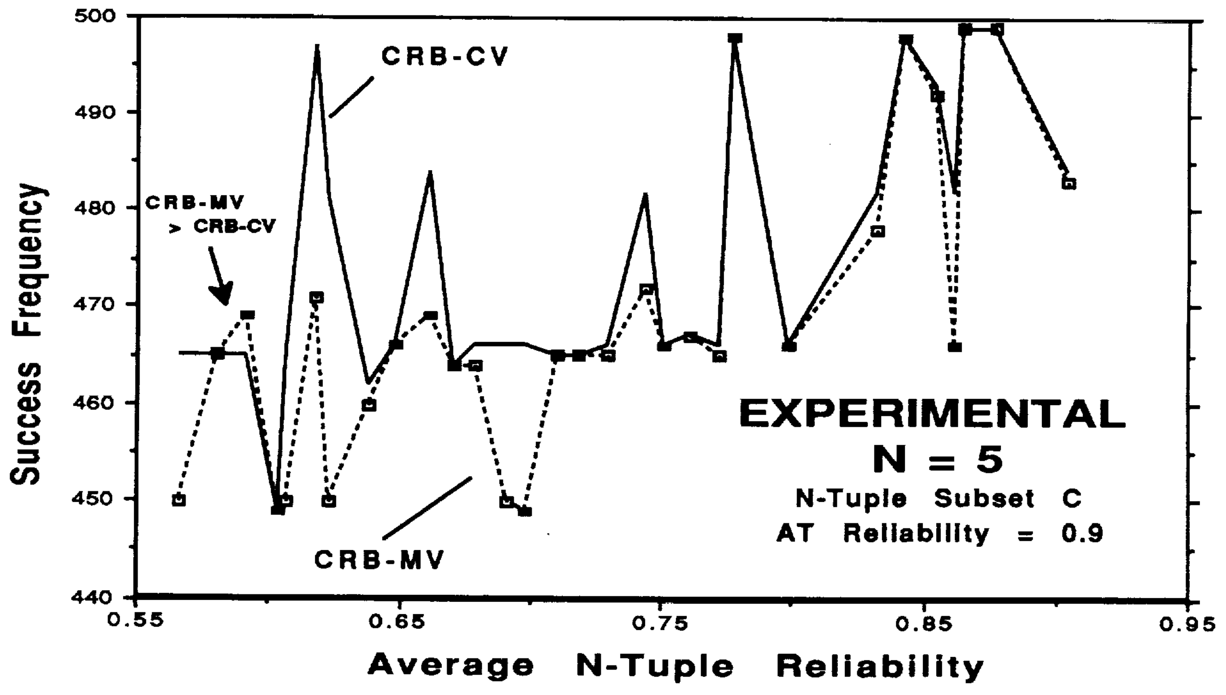

Figure 13. Comparison of CRB-MV and CRB-CV strategies.

Usually CRB-CV is more reliable than CRB-MV. However, if the number of agreeing versions is less than the majority sometimes the reverse may be true. For instance, if there is no majority then MV will fail and pass the decision to the acceptance test (which may succeed), while CV will vote and if the plurality is incorrect because of identical and wrong answers CV will return an incorrect answer. An example is given in Figure 13 and in columns 5 and 8 of Table 3.

A general observation regarding the behavior of CRB in the presence of failure correlation is that it is surprisingly robust. Our results appear to follow the general trends predicted by theoretical models based on failure independence.

3.4 Acceptance Voting

Acceptance Voting is very dependent on the reliability of the acceptance test (AT). If the AT is sufficiently reliable, AV can sometimes perform better than RB (e.g. columns 13 and 15 in Table 3). Also, reduction in effective output space cardinality may make RB and AV superior to CRB. In some situations AV performs better than CRB or any other voting based approaches (e.g. columns 14 and 15 in Table 3). Figures 14, 15 and 18 illustrate the theoretical behavior of AV in comparison with other strategies given failure independence and different AT reliability (β, cf. Figures 14 and 15) and output space cardinality (cf. Figures 15 and 18). But, in general, AV systems will be less reliable than CRB systems. Figures 16 and 17 illustrate experimental results for N=3 and 5 respectively.

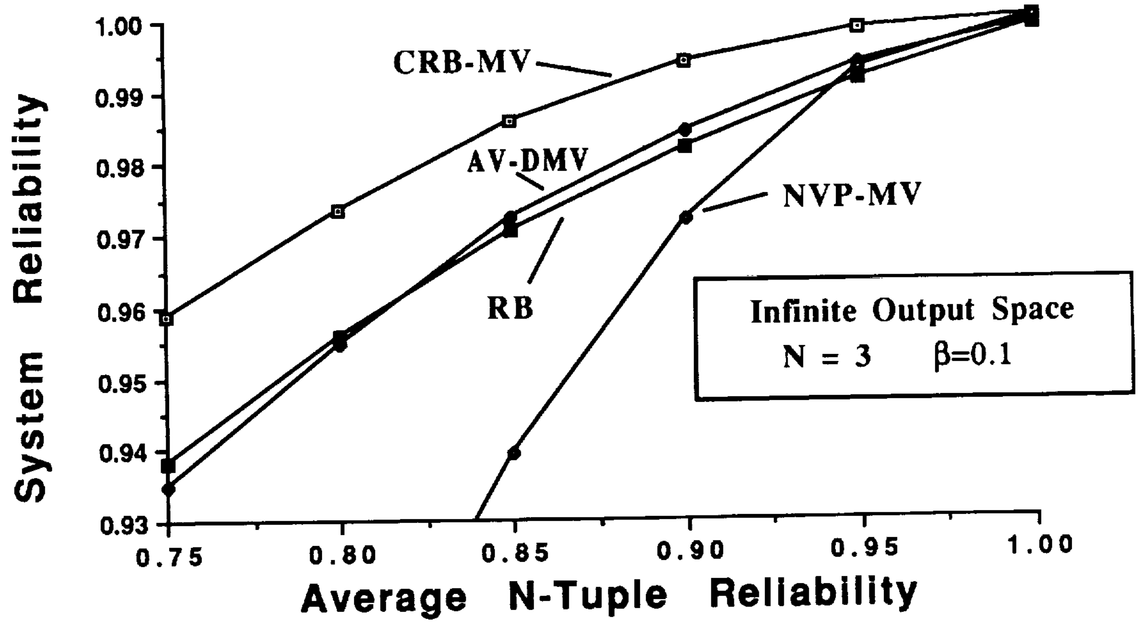

Figure 14. System reliability under Dynamic Majority Voting vs. version reliability assuming infinite cardinality of the output space for N = 3, and β = 0.1.

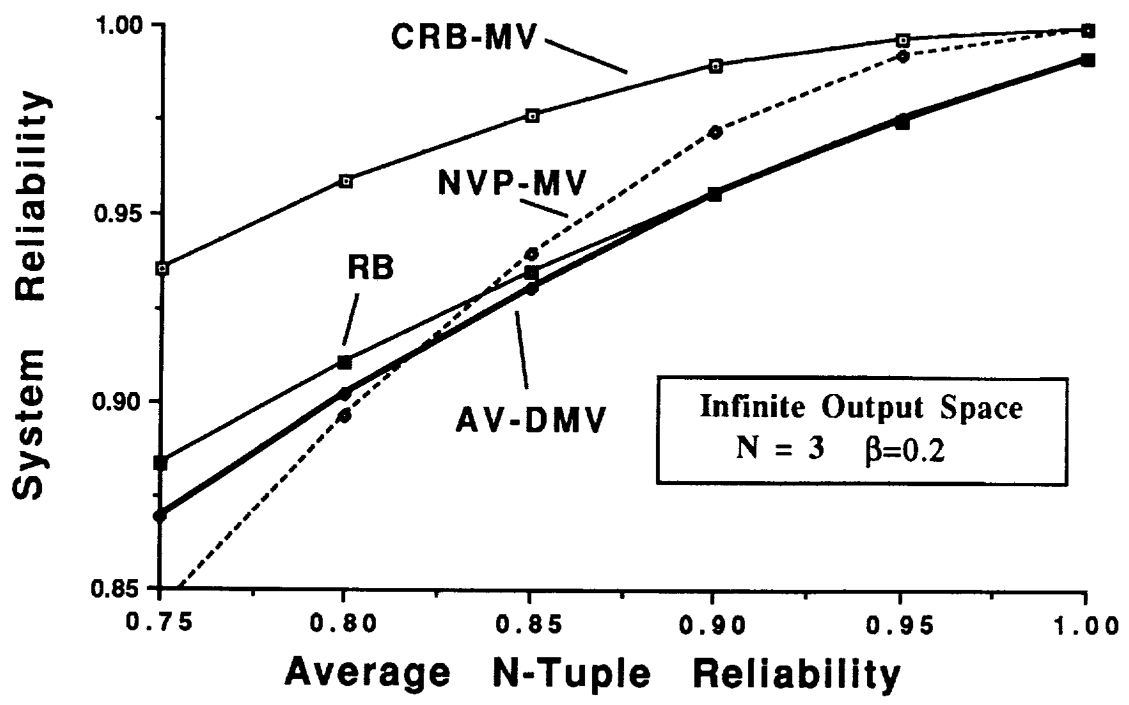

Figure 15. Influence of the output space cardinality

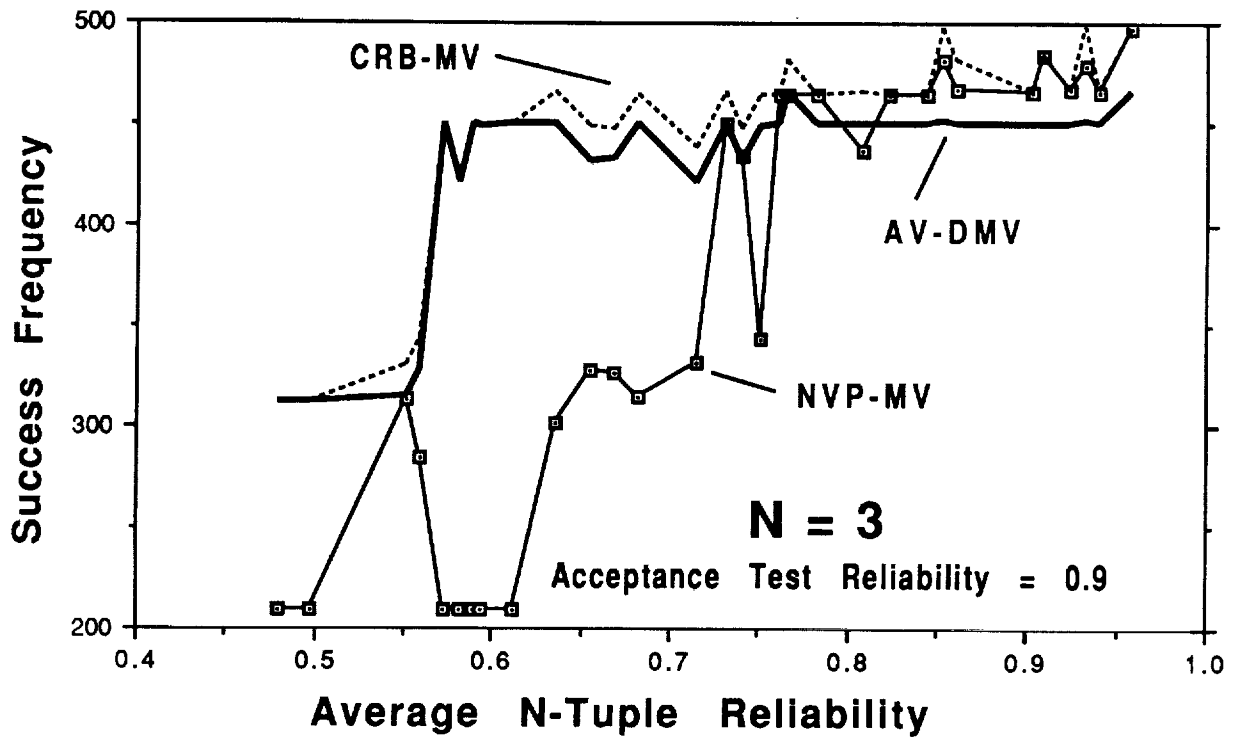

Figure 16. A comparison of CRB and AV, N=3.

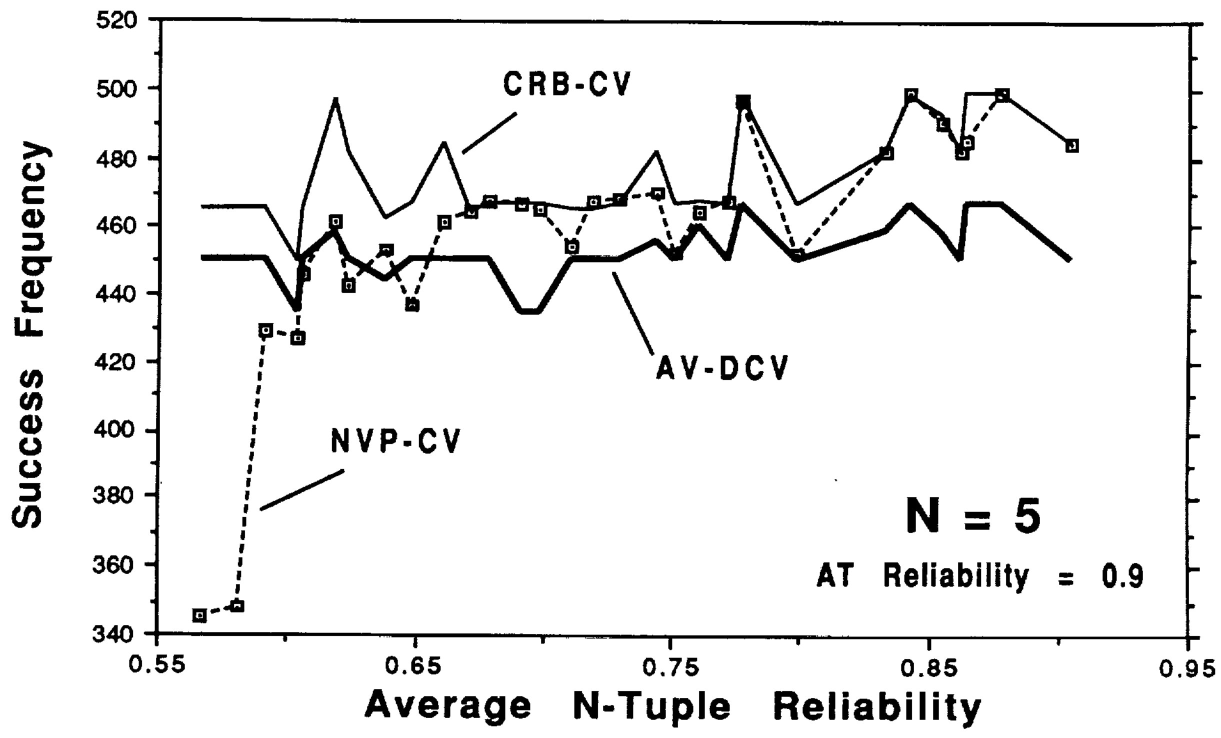

Figure 17. A Comparison of CRB and RV, N=5.

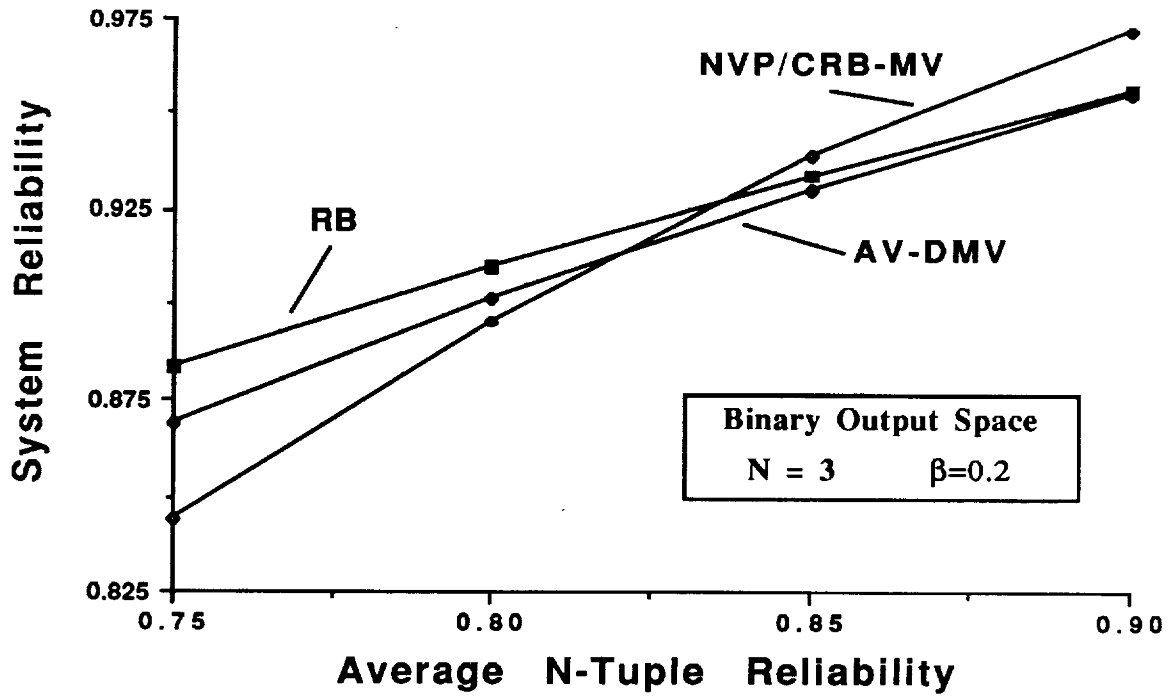

Figure 18. A comparison of RB, AV and CRB for low version reliabilities.

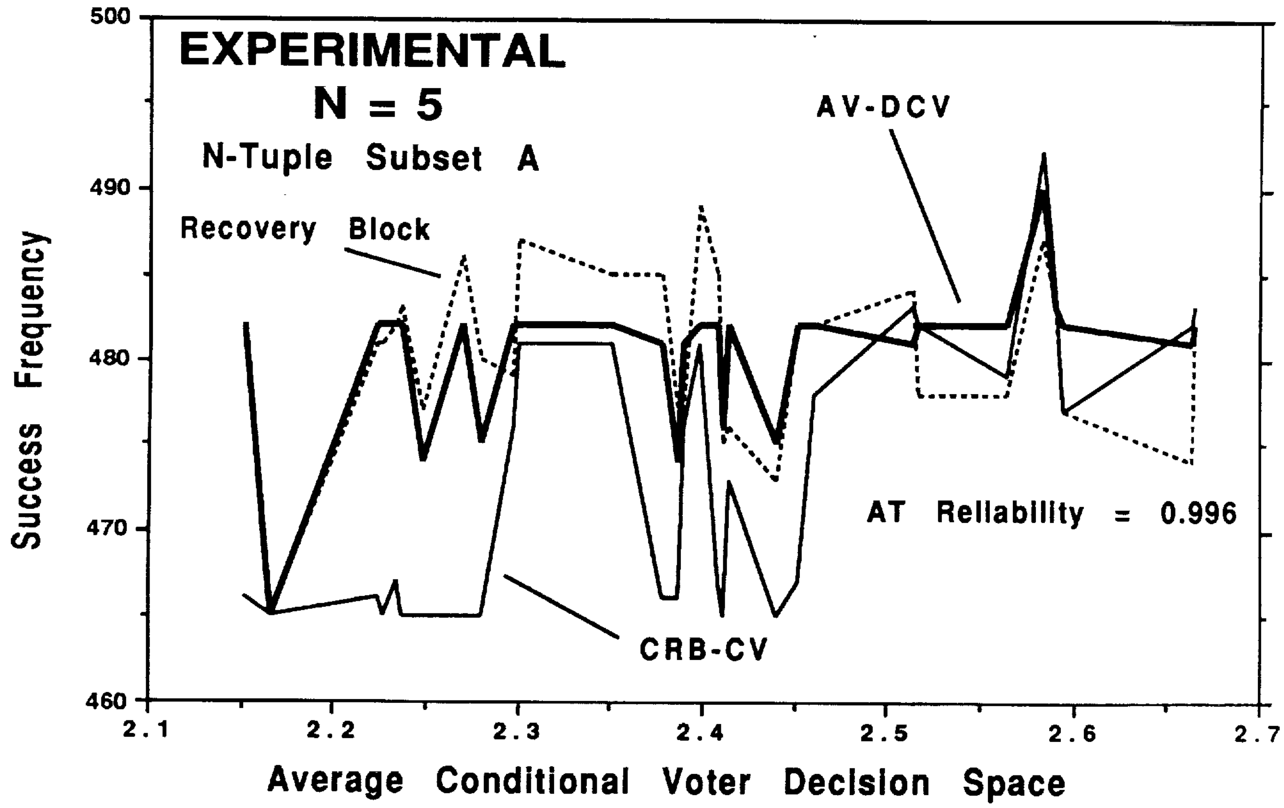

Figure 19. In small voter decision spaces AV may be superior to CRB.

AV reliability performance is usually superior when there is a large probability that a CRB voter would return a wrong answer, but at the same time the AT is sufficiently reliable so that it can eliminate most of the incorrect responses before the voting. The theoretical behavior of AV for a binary space using an AT with 0.8 reliability is shown in Figure 18. Figure 19 compares experimental behavior of AV and CRB for a range of conditional voter decision space cardinalities. We see that AV does indeed perform better than CRB-CV when voter decision space is small. Very highly correlated failures may still make AV perform as poorly as CRB, but in general in the case of very a small voter decision space AV tends to outperform CRB provided the AT is sufficiently reliable.

4. Summary and Conclusions

We used 20 independently developed but functionally equivalent software versions to investigate and compare empirically some properties of N-version programming, Recovery Block and Consensus Recovery Block using the majority and consensus voting algorithms. We have also compared this with another hybrid fault-tolerant scheme called Acceptance Voting using a dynamic versions of consensus and majority voting.

Consensus voting provides adaptation of the voting strategy to varying component reliability, failure correlation, and output space characteristics. Since failure correlation among versions effectively changes the cardinality of the space in which voters make decisions, consensus voting is usually preferable to simple majority voting in any fault-tolerant system.

In the presence of a large differences among version reliabilities, the version with the best reliability will perform better than any of the fault-tolerant techniques. Consensus Recovery Block produces the highest reliability most of the time. It outperforms N-version programming, but also very successfully competes with Recovery Block in situations where the acceptance test is not of the highest quality. Consensus Recovery Block is surprisingly robust even in the presence of failure correlation. Acceptance Voting is, under special circumstances, more reliable than both Consensus Recovery Block and Recovery Block. It, in general, has lower reliability than the others.

References

[Ath89] A.M. Athavale, "Performance Evaluation of Hybrid Voting Schemes," M.S. Thesis, North Carolina State University, Department of Computer Science, 1989.

[Avi77] A. Avizienis and L. Chen, "On the Implementation of N-version Programming for Software Fault-Tolerance During Program Execution", Proc. COMPSAC 77, 149-155, 1977.

[Avi85] A. Avizienis, "The N-Version Approach to Fault-Tolerant Software," IEEE Trans. Soft. Eng., Vol. SE-11 (12), 1491-1501, 1985.

[Bel90] F. Belli and P. Jedrzejowicz, "Fault-Tolerant Programs and Their Reliability," IEEE Trans. Rel., Vol. 29(2), 184-192, 1990.

[Eck85] D.E. Eckhardt, Jr. and L.D. Lee, "A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors", IEEE Trans. Soft. Eng., Vol. SE-11(12), 1511-1517, 1985.

[Kel88] J. Kelly, D. Eckhardt, A. Caglayan, J. Knight, D. McAllister, M. Vouk, "A Large Scale Second Generation Experiment in Multi-Version Software: Description and Early Results", Proc. FTCS 18, pp 9-14, June 1988.

[Kni86] J.C. Knight and N.G. Leveson, "An Experimental Evaluation of the assumption of Independence in Multiversion Programming IEEE Trans. Soft. Eng., Vol. SE-12(1), 96-109, 1986.

[Lit90] B. Littlewood and D.R. Miller, "Conceptual Modeling of Coincident Failures in Multiversion Software," IEEE Trans. Soft. Eng., Vol. 15(12), 1596-1614, 1989.

[McA90] D.F. McAllister, C.E. Sun and M.A. Vouk, "Reliability of Voting in Fault-Tolerant Software Systems for Small Output Spaces", to appear in IEEE Trans. Rel., December 1990.

[Ran75] B. Randell, "System structure for software fault-tolerance", IEEE Trans. Soft. Eng., Vol. SE-1, 220-232, 1975.

[Sco84a] R.K. Scott, J.W. Gault, D.F. McAllister and J. Wiggs, "Experimental Validation of Six Fault-Tolerant Software Reliability Models", IEEE FTCS 14,1984

[Sco84b] R.K. Scott, J.W. Gault, D.F. McAllister and J. Wiggs, "Investigating Version Dependence in Fault-Tolerant Software", AGARD 361, pp. 21.1-21.10, 1984

[Sco87] R.K. Scott, J.W. Gault and D.F. McAllister, "Fault-Tolerant Software Reliability Modeling", IEEE Trans. Software Eng., Vol SE-13, 582-592, 1987.

[Tri82] K.S. Trivedi, "Probability and Statistics with Reliability, Queueing, and Computer Science Applications, Prentice-Hall, New Jersey, 1982.

[Vou85] M.A. Vouk, D.F. McAllister, K.C. Tai, "Identification of correlated failures of fault-tolerant software systems", in Proc. COMPSAC 85, 437-444, 1985.

[Vou90] Vouk, M.A., Caglayan, A., Eckhardt D.E., Kelly, J., Knight, J., McAllister, D., Walker, L., "Analysis of faults detected in a large-scale multiversion software development experiment," Proc. DASC '90, pp 378-385, 1990.

Appendix I

To select subsets of N-tuples which have certain properties such as approximately equal reliabilities we use the following approach.

We first select acceptance test versions (for example, one low reliability, one medium and one high reliability acceptance test). These versions are then removed from the pool of 20 versions. Also removed from the pool might be versions which have either very low or very high reliability to better balance reliabilities of the selected N-tuples. For a given N the remainder of the versions are then randomly sampled until an N-tuple is formed, one version at the time, which has not already been accepted for the subset. The average N-tuple reliability is then computed, and if it lies within the desired reliability range the N-tuple becomes a member the subset. Once the subset contains either all possible combinations, or at least 600 N-tuples (whichever comes first), the subset is sorted by N-tuple reliability and standard deviation of the N-tuple reliability. If a single reliability category was used (e.g. between 0.8 and 0.9) then the first 30 versions with the smallest N-tuple standard deviation are chosen and run in the experiment. If a range of reliabilities is desired, the range is divided into into categories in such a way that members of the same category have identical first two digits after the decimal point. Then from each category we chose the combination that has the smallest standard deviation of the N-tuple reliability.

We have thus chosen a number of subsets. The following are mentioned in the text

N-Tuple Subset A (5 version systems, 5-tuple reliability in the range 0.8 to 0.9, acceptance test reliabilities of 0.67 (version 20), 0.93 (version 2) and 0.996 (version 18). Version 10 was not used (too reliable).

N-Tuple Subset B (3 version systems, 3-tuple reliability in the range 0.5 to 1.0, acceptance test reliabilities of 0.67 (version 20), 0.87 (version 3) and 0.90 (version 17). Version 10 was used.

N-Tuple Subsets C and D were chosen in a manner similar to set B except that they consisted of 5- and 7-tuples, respectively.