A NEW PARADIGM FOR THE DESIGN OF DEPENDABLE SYSTEMS

Behrooz Parhami

Dept. of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106, USA

ABSTRACT

A new general strategy for dependability procurement in digital systems that allows highly selective use of redundancy with low overhead is described. The proposed approach is based on attaching a dependability tag (d-tag) to each data object and updating pertinent d-tag values as the computation progresses. Normal operations on data objects tend to lower d-tag values while comparisons and voting on redundant copies of the same result, possibly obtained by different resources and/or algorithms, work in the opposite direction. Judicious intermixing of dependability-lowering and dependability-raising operations (driven dynamically by the dependability requirements for various results) can lead to desired dependabilities for computation results with minimal cost. Following an exposition of basic concepts of the proposed method with the assumption of perfect (error-free) d-tags and operations, a program of research is outlined for dealing with erroneous d-tags, imperfect operations, and other extensions. Some benefits and applications of the proposed approach are discussed in conclusion.

1. INTRODUCTION

Dependability is an important attribute of modern digital systems and of the computations performed on them. Even though the use of redundant resources for fault tolerance is an age-old design technique (e.g., the engineer's "safety factor"), systematic application of redundancy to the design of dependable digital systems did not start until the early 1960s when it was established that the reliability requirements of telephone switching systems and on-board spacecraft computers could not be met with conventional designs. Today's components are significantly more reliable than those used in early computers and design techniques (for hardware and software) are far more sophisticated. Yet the need for redundancy as a dependability procurement measure has not diminished. The main reasons for greater reliance on redundancy techniques are increased complexity and diversified applications in online and safety-critical systems. With the current trends toward more demanding and critical applications of digital systems by a growing number of minimally trained users, the dependability of computations becomes even more important.

In the 30-year history of dependable computing and through three generations of highly dependable systems, numerous techniques have been proposed for increasing the dependability of computations through the avoidance and/or tolerance of undesired conditions such as defects, faults, errors, malfunctions, and degradations that can potentially lead to result-level failures [5]. Such proposed methods of dependability procurement fall into two categories: (1) Structural methods and (2) Functional methods.

Structural methods employ static and dynamic hardware redundancy (among other schemes) to make the system highly dependable, so that any process run on it can be assumed to produce dependable result, given that the process itself is trustworthy. The resulting increase in dependability is universal and applies to every process or algorithm, regardless of whether or not this level of dependability is needed in all cases. Such processes or algorithms are either identical to the ones run on the corresponding non-redundant system or are modified versions that include checkpointing provisions. Clearly, this "worst-case" philosophy is neither efficient nor flexible for general use. Functional methods, on the other hand, are applied to specific processes or algorithms and take advantage of particular properties of the problem at hand to provide an explicit cost-dependability tradeoff. They are typically more efficient and/or selective in their use of redundancy but imply increased design effort for computations that deal with the specially modified data structures or to provide multiple versions of the required processes. Thus, for example, a process may be selectively replicated 2, 3, or 5 times for varying degrees of safety and/or resilience as required, provided of course that the correspondence between dependability and degree and type of replication is known (not a simple issue!).

In many situations, one can benefit from an even more selective approach to the application of redundancy, not only from one process to another but also within a single application process. For example, the degree of replication of a data object manipulated by a process may depend on:

1. Value or criticality of the data: Critical data must be protected through a higher degree of replication.
2. Ease of regenerating the data: Error detection with good coverage may be sufficient for easily regenerated data.
3. Size of the data object: High degree of replication is less desirable for large data objects.
4. Resilience of the data object: The higher the resilience, the lower the needed degree of replication.
5. Amenability to consistency check: Duplication and checking may be a substitute for greater replication.
6. Extent and type of use: A data object that is referenced rarely or in read-only mode may need less protection.

Unfortunately, incorporating various degrees of replication for data objects in each application process is hopelessly complex, even if all of the above aspects could be reasonably quantified. Furthermore, certain of these aspects (e.g., size of data object and its usage) cannot be accurately predicted as they may depend on run-time conditions. It follows that a general framework for handling varying degrees of replication along with a capability for automatically deciding on the required replication factors is needed.

2. A DATA-DRIVEN APPROACH

2.1. The Concept of Dependability Tags

Suppose that a dependability tag (d-tag) is attached to each data object as an indicator of the data object's correctness probability. Thus, a data object D and its d-tag d will comprise a composite object $\langle D, d \rangle$. The d-tag d assumes values from a finite set of dependability designations $d \in \{0, 1, \dots, \delta-1\}$, where δ is an application-dependent constant. Associated with each d-tag value d, are constants π_d and π'_d such that the d-tag d in $\langle D, d \rangle$ signifies:

$$\pi_d \le \text{prob}[D \text{ is correct}] \le \pi'_d$$

We will assume that $\pi_j < \pi_{j+1}$ and $\pi'_j \le \pi'_{j+1}$, so that a larger d-tag value implies higher confidence in the correctness of the associated data object. Unless otherwise stated, the upper bound π'_j is assumed to be 1 in the remainder of this paper. We also assume $\pi_0 = 0$, $\pi_{\delta-1} = 1$ (d-tag values of 0 and $\delta - 1$ reserved for hopeless and perfect values, respectively), as well as randomness and statistical independence of errors in various data objects.

Note that as defined, d-tags essentially represent a general and very flexible discretization scheme for correctness probabilities. In other words, the π_j values need not be selected to conform to any particular rule or pattern. However, in practice, it is desirable to have a capability for greater discrimination at the high end of dependability values. This is because correctness probabilities 0.99 and 0.999 are significantly different while the values 0.4 and 0.5 need not be distinguishable as they both represent practically useless values. Although theoretically it is possible to associate d-tags with data objects at any level, practical considerations such as data storage redundancy and computational overhead will probably restrict meaningful applications to high-level data objects with complex structures and operations. Of course, regardless of the level at which d-tags are applied, the problem of determining d-tag values is non-trivial and must be dealt with in depth. However, given correctly assigned d-tags and ignoring for now the possibility that the d-tags themselves may be corrupted in the course of our computations, we can discuss the manipulation of tagged data in terms of dependability-lowering and dependability-raising operations.

2.2. Dependability-Lowering Operations

Normal operations on data objects tend to lower the d-tag values. Assuming that operations are themselves perfectly dependable, the dependability of each result is only a function of the operands' dependabilities. A unary operator u tansforms the data object D into $u(D)$. In our scheme, we define for each u, a unary operator u^* such that $u^*(\langle D , d\rangle) = \langle u(D) , d\rangle$. This simply means that the dependability of the result $u(D)$ is the same as the dependability of the operand D. In the case of a binary operator b in $D = b(D', D'')$, we define the corresponding operator b^* which operates on composite data objects giving $\langle D, d\rangle = b^*(\langle D', d'\rangle,\langle D'', d''\rangle)$. The key to this extension is a procedure for determining d from d' and d''. Thus

$$b^*(\langle D' , d'\rangle , \langle D'' , d''\rangle) = \langle b(D', D'') , g(d', d'')\rangle$$

where g is the dependability evaluation function associated with binary operators. More generally, one can consider a dependability evaluation function $g_b(d', d'')$ for each binary operator b or even $g_b(d', d'', D', D'')$. However, let's keep things simple for now.

Assuming that $b(X,Y)$ depends on both X and Y (i.e., b is not actually a unary function) we can define $g(d', d'')$ as follows:

$$g(d', d'') = d \quad \text{such that} \quad \pi_d \le \pi_{d'}\pi_{d''} < \pi_{d+1}$$

Because of the way d-tags are defined, the value of d thus obtained satisfies $d \le \min(d', d'')$. The above can easily be generalized to a k-variable function by using a recursive definition. At any rate, the dependability of the result is never more than the smallest d-tag value involved. This is why we call all such operations "dependability-lowering" operations. If the final data objects (computation results) end up with acceptable d-tags after all such lowerings, then nothing more needs to be done. Otherwise, we need to structure the computation in such a way that it also includes dependability-raising operations at some points.

2.3. Dependability-Raising Operations

Suppose that we obtain a result in two different ways using hardware, software, or time redundancy. Let the two results and their corresponding d-tags be $\langle D_0,d_0\rangle$ and $\langle D_1,d_1\rangle$. We wish to draw the conclusion that the result is $\langle D,d\rangle$ where d is the highest possible d-tag that can be attached to a value D inferred from the inputs. Obviously, if $D_0 \ne D_1$, then $\langle D,d\rangle = \langle D_i,d_i\rangle$, $i \in \{0,1\}$, such that $d_i \ge d_{1-i}$. If $D_0 = D_1$, then $D = D_0 = D_1$ and d is computed as follows. Let p_0 and p_1 be the actual correctness probabilities for D_0 and D_1, respectively. By definition, $p_i \ge \pi_{d_i}$ for $i = 0,1$. Then the probability that D is correct can be written as:

$$p = p_0p_1 / [p_0p_1 + (1 - p_0)(1 - p_1)]$$

The value of p given above is a nondecreasing function of p_0 and p_1. Thus:

$$p \ge \pi_{d_0}\pi_{d_1} / [\pi_{d_0}\pi_{d_1} + (1 - \pi_{d_0})(1 - \pi_{d_1})]$$

To maximize the d-tag of the result, d must be selected such that:

$$\pi_d \le \pi_{d_0}\pi_{d_1} / [\pi_{d_0}\pi_{d_1} + (1 - \pi_{d_0})(1 - \pi_{d_1})] < \pi_{d+1}$$

Even though these inequalities do not guarantee that d will be higher than d_0 or d_1, judicious choice of the constants π_j can ensure that in all practical cases dependability is raised to at least j+1 when both inputs have d-tags j.

The above is easily generalized to n versions of a result. Let the n versions be $D_0, D_1, \ldots , D_{n-1}$, with the corresponding d-tags $d_0, d_1, \ldots , d_{n-1}$. If all of these data objects are different, then the output is taken to be $\langle D_i,d_i\rangle$ such that $d_i \ge d_j$ for $0 \le j < n$. At the other extreme, if all of the n data objects are identical, then the output is set to one of them and assigned the d-tag value d such that:

$$\pi_d \le (\Pi_{i=0..n-1}\pi_{d_i}) / [\Pi_{i=0..n-1}\pi_{d_i} + \Pi_{i=0..n-1}(1-\pi_{d_i})] < \pi_{d+1}$$

In general, the n data objects can be partitioned into classes of identical objects. For each class, a d-tag value is computed and the class with the largest associated d-tag value determines the output.

3. A PROGRAM OF RESEARCH

3.1. Formalization and Refinement

Several issues arise in selecting the discretization probability thresholds π_j, $0 \le j \le \delta - 1$ for practical application of d-tags. Clearly, the value of δ affects both the storage and processing overheads for dealing with d-tags. The optimal value of δ for each application must initially be selected on the basis of estimates for these overheads and the requirements of dependability-lowering and dependability-raising operations. Intuitively, a larger value of δ, with the correspondingly finer subdivisions defined by π_j, causes smaller reductions in d-tag values while a smaller δ may yield potentially larger increases in comparison-type operations. Initial research has shown that there are values of δ with associated values for π_j that satisfy these two conflicting requirements simultaneously.

In general, a range of values for δ satisfies the requirements of dependability-lowering and dependability-raising operations. The smaller values in the acceptable range obviously imply lower direct storage and processing overheads. However, selecting a larger value for δ may eventually translate into lower replication requirements for data and computations in view of the higher likelihood of smaller decreases in d-tag values and thus a potentially lower overall cost. To be able to properly evaluate such tradeoffs, the concept of d-tags, and their associated direct and indirect costs, must be formalized and further refined.

3.2. Dealing with Imperfect d-Tags

Obviously, d-tags are themselves subject to errors and cannot be trusted completely. An erroneous d-tag may be lower or higher than the correct value. In the first case, the error is "safe" in the sense that at the end, a *negative d-tag error* can lead to one of three things:

1. A correct result with erroneously low, but still acceptable, d-tag value that is trusted and used.
2. A correct result with unacceptably low d-tag value that is either discarded or used cautiously.
3. An incorrect result with correct d-tag that is either discarded or used with an appropriate level of care.

The third possibility arises because erroneously low d-tag values for correct results may cause incorrect values to prevail in comparison-type operations. However, such incorrect values will carry with them appropriate d-tags that indicate their dependability, provided that no error of the second type occurs.

(217) 333-4789 Further information can be obtained from Dr. W. Kenneth Jenkins.

It is, therefore, sufficient to guard against errors that improperly raise the d-tag values. Such *positive d-tag errors* may have three causes:

1. Incorrect storage and/or transmission of d-tag values.
2. Errors during dependability-lowering operations.
3. Errors during dependability-raising operations.

Storage and transmission errors are the easiest to deal with. One can of course use error codes in a straightforward manner for protection. However, several properties of this particular application can be exploited to devise more effective methods. The fact that we are only interested in protecting against positive d-tag errors suggests that any asymmetry of errors (e.g., higher likelihood of 1-to-0 compared to 0-to-1 errors) can be exploited by proper encoding. Also, since the arithmetic value of a d-tag is significant, arithmetic error codes are potentially useful. These can be combined with a gray-like encoding scheme to limit the damage caused by an uncaught error.

Determination of d-tag values during dependability-lowering and dependability-raising operations essentially involves simple function evaluations. Assuming that these functions are evaluated by table lookup, many different methods can be used to detect and correct potential errors. For example, with 4-bit d-tags, a 256-word by 4-bit table will be needed in a binary operation. For protection against errors, one can store the two 4-bit tags with each 4-bit table entry and then encode the resulting 12-bit entries in some error code. Alternatively, the use of self-checking circuits for the manipulation of d-tags may be contemplated.

3.3. Dealing with Imperfect Operations

Even when the data objects D' and D'' are perfectly dependable, the data object $D = b(D', D'')$ may have potential errors due to imperfect hardware or software implementing the binary operation b. To deal with this problem, appropriate dependabilities must be assigned to various operations and used in determining the d-tag values for operation results. The key issue here is to use dependability estimates that are pessimistic (so that any resulting d-tag error is on the safe side) but not too pessimistic so as to require excessive redundancy to overcome their effects.

Comparison and voting operations may also be imperfect. For example, in the comparison of two versions of a result giving $\langle D,d\rangle = c(\langle D1,d1\rangle,\langle D2,d2\rangle)$, an error may occur in judging the equality of $D1$ and $D2$. If $D1 = D2$ but it is erroneously determined that they are unequal, the error is on the safe side in that dependability is not raised. If, on the other hand, $D1 \neq D2$, dependability may be erroneously raised by a comparison error. Therefore comparison mechanisms (hardware or software) must be designed to have asymmetric error modes. Similar observations apply to voting errors when more than two copies of a data object are involved. Obviously, the criticality of the method for dealing with imperfect operations increases directly with any increase in the dependability of original data sources.

Imperfect operations on tagged data objects can be modelled in several ways. One possibility is to view the operation as simply another element of the computation with its associated d-tag. Then, for example, a binary operation with the associated d-tag d_b performed on data objects having d-tags d_1 and d_2 will be like a perfect ternary operation on data objects having the d-tags d_b, d_1, and d_2. This conceptually simple scheme does not solve all of our problems as the assignment of d-tags to various operations is nontrivial. Another possibility is to attach to each data object a second "operation count" tag. Then, assuming that all operations have roughly equal complexities, the operation count tag provides a second indication of how dependable the data object is by showing how many transformations it has undergone. Operation count tags can be periodically reset to zero and their effects incorporated into d-tags by suitable adjustment algorithms.

3.4. Evaluation and Extensions

Although the development of mathematical models for evaluating the cost-effectiveness of the proposed approach should proceed in parallel with the resolution of the above problems, it is still necessary to integrate the various aspects of the modeling process into a "clean" mathematical model and to evaluate and fine-tune the proposed techniques. An aspect of this evaluation is the identification of application areas where the proposed approach is likely to lead to improved cost-effectiveness compared to current design paradigms and to formulate a research plan for dealing with these application areas in more detail.

Once the promising application areas for data-driven dependability procurement have been identified, design issues must be examined in greater detail for selected areas in order to develop verifiable quantitative results on the gains and benefits versus the accrued cost. It may also be desirable to examine the areas where data-driven dependability procurement appears to be impractical or uneconomical to discover the underlying reasons and to look for extensions and/or variations of the technique (or combinations with other methods) that may overcome the perceived problems. There is also a potential for using the data-driven dependability procurement paradigm to build a unified framework for the automatic synthesis of ultrareliable systems. Some ideas along this line are presented in Subsection 4.5. Given that such a framework can be established, reduction of the design time and cost will appear as a new factor in comparing the cost-effectiveness of various approaches and may well tip the balance in favor of the data-driven dependability procurement approach in additional application areas.

4. BENEFITS AND APPLICATIONS

In this Section, we will enumerate and briefly discuss certain areas where the application of data-driven dependability procurement with d-tags seems promising or where it provides a beneficial way of looking at old problems and techniques. The examples in this section are primarily based on very simple data objects although as stated earlier, practical considerations will probably restrict meaningful applications to higher-level data objects.

4.1. Data Diversity and d-Tags

Take the simple example of computing the area of a triangle. For this computation, it is sufficient to know two sides and the angle between them. Suppose that we know all three sides and all three angles, each with its associated d-tag. Then, various subsets of the available data can be used with different procedures (formulas) to obtain the desired result. Each result will have an associated d-tag. At the end, we can either select the result with the highest d-tag value or try to combine the various results into a single result through some form of inexact comparison and/or voting. The difference between the above suggested method and the data diversity scheme as defined by Amman and Knight [1] is that in the latter, the computation is always performed with the primary data first; alternate or "reformulated" data enter the picture only if an acceptance test is not satisfied. Because acceptance tests seldom provide perfect coverage, it may be advisable to perform the computation with several sets of data before making the final decision. Dependability tags provide a convenient mechanism for keeping track of potential errors and inaccuracies and for algorithmically combining a multiplicity of potentially incorrect results.

4.2. Design Diversity and d-Tags

Consider the "consistent comparison problem" defined by Brilliant, Knight, and Leveson [3]. The problem, simply stated, is as follows. If multiple versions of a system with diverse designs [2] are used for performing an inexact computation, slight computation errors do not necessarily produce correspondingly small inaccuracies in the final results of the versions. One reason is that when a version reaches a decision point where one of two algorithm paths must be selected based on a comparison of inexact values that happen to be nearly equal, either path may be selected with non-negligible probability and the result becomes unpredictable. We can solve this problem through the use of d-tags if each comparand has a d-tag that reflects its accuracy as well as dependability and if the d-tag for the comparison result is made a function of the relative difference of the two values; i.e., the comparison result is assigned a high d-tag value if one operand is much larger than the other and a very low d-tag value if the two are nearly equal. When such a comparison result becomes the basis for a path selection, its dependability will affect the d-tags of all data that are dealt with subsequently. Again, routine combination of multiple results and their d-tags yields the final result along with an indication of its dependability.

4.3. Dependable Dataflow Systems

The dataflow model of computation [7] is particularly supportive of the use of data-driven dependability procurement with d-tags. One can envisage data tokens carrying one or more dependability-related tags that are manipulated according to specific rules as the computation unfolds. At the very end, results will have dependability indicators attached to them. If in the course of a computation, dependabilities fall below a certain threshold, program segments holding backup or alternate computations are activated in order to check or verify the obtained results. Such verifications will raise the dependability values and will allow the computation to proceed. In the demand-driven variant, much effort can be saved by producing results that are not overly dependable. This claim may seem strange at first but suppose that we want to compute $f(g(D), h(D))$, where f, g, and h are arbitrary functions and D represents a data object. The computation starts by generating a demand for the value of f with a certain d-tag value d_f. This will in turn generate demands for the evaluation of g and h with d-tag values d_g and d_h that are just high enough to ensure that the final result will have the desired dependability. Finally, the demand for the values of g and h will generate a demand for the data object D will a certain dependability d. In general, several sets of values for d_g and d_h may satisfy the final dependability requirement and the selection of optimal values is non-trivial. However, the concept is quite useful and worthy of further research.

4.4. Dependable Distributed Systems

In a distributed computing environment, d-tags can be used in many different ways. Consider for example a distributed database consisting of various data objects (e.g., relations in the relational database model), each with multiple copies stored at different sites. Each processor may have copies of several data objects with their associated d-tags. Exchanging information about the data objects between processors can modify the d-tag values dynamically, restoring the gradually deteriorating dependabilities to acceptable levels and correcting erroneous information which may have aquired high d-tag values due to malicious faults. Developing the detailed protocols and decision algorithms for this class of applications constitutes a fruitful area for further investigation. Note in particular that d-tags provide a convenient mechanism for implementing probabilistic agreement protocols. Each processor can be viewed as working on the construction of an "agreement" data object that it initializes with its own view of the world and a low associated d-tag value. As information is exchanged between processors, "agreement" data objects gradually acquire higher associated d-tag values. The process of information exchange, which itself is driven by the d-tags, continues until desired dependability levels are reached.

4.5. Unification of Concepts

As an added benefit, the data-driven dependability procurement paradigm can provide a framework for the unification and extension of several concepts in the field of dependable computing and contribute to a better understanding of the underlying principles of the field. Consider, for example, the two extremes of binary (two-valued) and continuous (real-valued) d-tags. With binary d-tags, voting on multiple copies of a result is achieved by ignoring all results that have d-tags of 0. This is equivalent to the self-purging redundancy scheme [4]; a generalization of hybrid redundancy which itself is a combination of voting and standby redundancy techniques. With binary results (bits as data objects) and real-valued d-tags denoting exact correctness probabilities (rather than a lower bound on them; i.e., $\pi_j = \pi_j'$) voting on multiple copies is equivalent to the use of optimal adaptive vote-takers [6]. The proposed data-driven approach can also lead to the development of sophisticated "voting" schemes for non-atomic data objects. Simple voting is based on retaining the matching majority and discarding the non-matching minority. However, for complex data objects, this approach is overly pessimistic because it invalidates a non-matching data object in its entirety. Component-by-component voting may be impossible because objects may be incomplete or non-identical in structure. Again d-tags provide a mechanism for keeping track of error probabilities and for algorithmically combining several potentially incorrect values into dependable final results.

5. CONCLUSION

Much of the literature on dependable computing is built on the binary distinction between correct and incorrect data. Thus, any data object, such as a word or a record, is either completely dependable or totally incorrect. Various levels of data accuracy are sometimes taken into account, but accuracy is quite different from dependability. Whenever "gray-level" dependability is considered, it is with respect to functional units that generate or transform and channels that carry the data, rather than the data objects themselves. Clearly, different data objects, or even diverse representations of the same object, can have different "error characteristics" when manipulated by identical functional units or transmitted over identical channels. Thus, the desirability of a data-driven approach.

In a way, the data-driven dependability procurement method is nothing but replication plus comparison and/or voting. The major difference with existing methods of dependability procurement based on the same concepts is that the replication factors and the invocation of voting and consistency check procedures are dynamically controlled by the d-tags associated with various data objects. The replication factor may be not only a function of the particular data object (structure and contents) but also dependent on the access method and frequency for a given critical computation. Although custom design based on a-priori estimates of data object size and access patterns may be effective in some cases, it is both expensive and ineffective when there are wide dynamic variations in the relevant parameters. Dependability tags, as defined here, provide a convenient mechanism for keeping track of potential errors and for algorithmically combining a multiplicity of potentially incorrect data into dependable results.

At least two application areas will greatly benefit from the proposed method. In the area of dataflow systems, where dependability studies have been few, sketchy, and inconclusive, data-driven dependability procurement is a natural method and is expected to lead to the a satisfactory approach for ultrareliable dataflow computation. For distributed computer system, important theoretical contributions have established the difficulty of the dependability procurement problem but few have led to practical design strategies. The use of d-tags (along with suitable distributed manipulation algorithms) is likely to lead to a solution to this problem as well.

REFERENCES

[1] Ammann, P.E. and J.C. Knight, "Data Diversity: An Approach to Software Fault Tolerance," *IEEE Transactions on Computers*, Vol. 37, No. 4, pp. 418-425, Apr. 1988.

[2] Avizienis, A. and J.P.J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments,"*Computer*, Vol. 17, Aug. 1984, pp. 67-80.

[3] Brilliant, S.S., J.C. Knight, and N.G.Leveson, "The Consistent Comparison Problem in N-Version Software," *Software Engineering Notes*, ACM SIGSOFT, Vol. 12, No. 1, pp. 29-34, Jan. 1987.

[4] Losq, J., "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," *IEEE Transactions on Computers*, Vol. C-25, No. 6, pp. 569-578, June 1976.

[5] Parhami, B., "From Defects to Failures: A View of Dependable Computing," *Computer Architecture News*, ACM SIGARCH, Vol. 16, No. 4, pp. 157-168, Sep. 1988.

[6] Pierce, W.H., "Adaptive Decision Elements to Improve the Reliability of Redundant Systems," *IRE International Convention Record*, Mar. 1962, pp. 124-131.

[7] Veen, A.H., "Dataflow Machine Architecture," *Computing Surveys*, Vol. 18, No. 4, pp. 365-396, Dec. 1986.