ANALYSIS OF THE IMPACT OF ERROR DETECTION ON COMPUTER PERFORMANCE

Kang G. Shin and Yann-Hang Lee
Department of Electrical and Computer Engineering
The University of Michigan
Ann Arbor, Michigan 48109, U. S. A.

ABSTRACT

Conventionally, reliability analyses either assume that a fault/error is detected immediately following its occurrence, or neglect damages caused by latent errors. Though unrealistic, this assumption has been imposed in order to avoid the difficulty of determining the respective probabilities that a fault induces an error and the error is then detected in a random amount of time after its occurrence.

As a remedy for this problem, in this paper a model is proposed to analyze the impact of error detection on computer performance under moderate assumptions. Error latency - the time interval between occurrence of an error and the moment of error detection - is used to measure the effectiveness of a detection mechanism. We have used this model to (1) perdict the probability of producing an unreliable result, and (2) estimate the loss of computation due to fault and/or error.

The work reported there is supported in part by NASA grant No. NAG 1-296. All correspondence should be made to Prof. K. G. Shin, ECE Dept., The University of Michigan, Ann Arbor, MI, 48109.

1. INTRODUCTION

During the past decade, many reliability-related models for fault-tolerant computers have been developed. Based on system structures and operation strategies, these models predict various measures such as reliability, computation capacity, performability, etc. Usually, in these models, a probability distribution function is used to describe the occurrence of component or system failure. The results represent the time-varying characteristics of a computer system. Since only the occurrence of failure is included in these models, they fail to cover the following two aspects. One is the existence of a latent fault in which case a fault is present but no erroneous state is induced. The other is the possible error latency because the error may not be detected immediately following its occurrence.

Consider the property of a fault. An input signal to a computer may cause the fault to induce some erroneous states, or it may simply pass through this fault and produce a correct output. The fault is said to be *latent* if it does not harm normal operations. Bavuso, et al., investigated the problem of latent fault and proposed experiments to measure fault latency [1]. Their studies indicate that a significant proportion of faults remained latent after many repetitions of a program. This fault latency has an important impact on an ultra-reliable system since it may cause a catastrophe if more than one latent fault becomes active.

It is desired that the error detection mechanisms associated with the system identify an error immediately upon generation. In fact, some errors may not be captured by error detection mechanisms when it occurs and then spread as a result of subsequent flow of information. Thus the damage by the error will be propagated until it is identified. The delay between the occurrence of an error and the moment of detection, called *error latency*, is important to damage

assesment, error recovery, and confidence in computed results. The same notion has been defined by Courtois as detection time [2,3] and by Shedletsky as latency difference [4]. Courtois also showed the results of on-line testing of the M6800 microprocessor and presented the distributions of detection time for cretain detection mechanisms. Shedletsky proposed a technique to evaluate the error latency based on the "fault set" philosophy and the probability distribution of input signals. The resultant error latency was used to decide the required rollback distance for successful data restoration. Both of these are confined to the study of error detection capability and are not extended to include the impacts of error detection on the system performance.

In reconfigurable fault-tolerant systems, a task executed by processors can be recovered through various recovery methods if one of the resident processors fails. Thus these systems are considered to have failure only when all resources are exhausted or the system fails to reconfigure. In practice, in addition to the probability of system failure, one may question what would happen if the system can not respond to a fault/error immediately following its occurrence. With the existence of error latency, the system may send out some erroneous computation results if it is still unaware of the error at the output phase. On the other hand, even if the system has detected the error before it is propagated, the computation achieved during error latency is useless and the whole system suffers from the delay caused by error latency and recovery. So the total cost induced by fault and/or error consists of two parts: one is the computation loss which includes error detection overhead, error latency, and recovery overhead; the other part is the relative cost increased due to delayed response. To quantify these effects of error latency, the probability of having an unreliable result and the computation loss have to be evaluated.

Various error detection techniques can be used to reduce the computation loss and enhance the reliability of computation results; for instance,

enhancement of self-checking capability so that most of errors can be detected immediately, limitation of the error contaminations to reduce error latency and recovery overhead, and periodic diagnostics which can seize faults before they induce errors. Each of these techniques by itself may not provide acceptable solutions to the reliability problem without high cost or overhead. Instead, a combination of these techniques must be employed to obtain a good, reliable performance at a reasonable cost.

In this paper, a model is proposed to describe error detection processes and to estimate their influences on system performance. Because intermittent faults can seriously degrade performance and can cause a large fraction of all errors [5], the model is intended to study their impact. In the following section, the classification and the properties of error detection mechanisms are discussed first. The model is developed in Section 3. Section 4 presents the evaluation of the probability of having an unreliable result and the estimation of average computation loss. The optimal strategy of periodic diagnostic is also discussed in this section. A brief conclusion follows in Section 5.

Note that in this paper we consider faults in hardware components which may cause a transition to erroneous states during the normal operation. We also assume that there is no design fault in the system. An error is defined to be the consequent erroneous information/data caused by fault(s).

2. CLASSIFICATION OF ERROR DETECTION MECHANISMS

There are various error detection mechanisms which can be incorporated in a computer system. The basic principle of these mechanisms is the use of redundancy in devices, information, or time. Based on where these mechanisms

are employed and their respective performance measures, they are divided into the following three classes.

1. *Signal level detection mechanisms*

Usually, the mechanisms in this category are implemented by built-in self-checking circuits. Whenever an error is caused by a predescribed fault, these circuits detect the malfunction immediately even if the erroneous signal does not have any logical meaning. Typical methods include error detection codes, duplicated complementary circuits, matcher, etc. Since the error is induced only when an input signal falls into the corresponding fault set of the fault, the fault latency will depend on the type of fault and the distribution of input signal. On the other hand, the error is detected immediately whenever it is generated. These detection mechanisms are difficult to have complete detectability for all kinds of error because (1) it is prohibitively expensive to design detection mechanisms which cover all types of faults, and (2) physical dependence between function units and detection mechanisms cannot be totally avoided. The performance of these detection mechanisms is measured by "coverage", which is the probability of detecting an arbitrary fault.

2. *Function level detection mechanisms*

The detection mechanisms in this level are intended to check out unacceptable activities or information at a higher level than the previous category. These detection mechanisms could be imagined as "barriers" around the normal operations. After an error is generated by a fault, the resulting abnormality may grow very quickly which is called "snow ball effect" [3], or "error rate phenomenon" [6], until it hits these barriers. We can apply several software and hardware techniques such as capability checking, acceptance checking, invalid op-code, timeout, and the like. Compared with the mechanisms in the first

category, these detection mechanisms are more flexible and inexpensive but the error latency tends to increase. The effectiveness of these detection mechanisms is very difficult to evaluate since it depends not only on the program executed and the current system states, but on the type of error.

3. *Periodic diagnostic*

This method is usually referred to as off-line testing because the computation unit can not perform any useful task while it is applied. It is composed of a diagnostic program which imitates inputs such that all existing faults are activated and thus produce errors. Several theoretic approaches exist to determine the probability of finding an error after a certain amount of test time (equivalent to the probability of detecting fault in this case) [7,8]. Tasar also provided a simulation to show the coverage of a self-testing program [9]. All these results indicate tnat the effectiveness of the present category is a monotonically increasing function of run time. Since the time required for complete testing is too long, it is impractical to apply this method frequently during normal operation. An alternative is to perform an imperfect diagnostic periodically during normal operation or perform a thorough diagnostic when the system is idle.

3. MODEL DEVELOPMENT

We have developed a model for describing error detection mechanisms as in Figure 2. The model consists of three parts: the occurrence of a fault, the consequent generation of an error, and the detection of that error. Since the probability of having a double fault at a time is negligible, the case of multiple faults is

excluded from this model. There are six states in the model as follows:

1). NF (non-faulty): In this state no fault exists in the system.

2). F (faulty): There is a fault which is active and capable of inducing errors.

3). FB (fault-benign): There is an inactive intermittent fault.

4). E (error): There is at least one error in the system and the fault which has yielded erroneous information is still present.

5). ENF (error-non-faulty): In this state the intermittent fault has become inactive after it induced some errors.

6). D (detection): In this state, the detection mechanisms have identified some errors in the system.

Usually, the occurrence of a fault is regarded as a Poisson process with rate λ. Since the system may contain an inactive intermittent fault, a benign state has to be included in the model. Several models of intermittent faults were proposed and used for testing and reliability evaluation [10 - 15]. In our model, the transitions between states NF, F, FB, and between states E, ENF are used to describe the behavior of faults.

Suppose there exists a process for generating errors by a given fault. With the assumption that the signal patterns in successive inputs are independent, the period of fault latency can be considered to be a random variable with a hyperexponential distribution (or composite geometric distribution for discrete inputs or cycles [8]). Using the concepts of information theory, Agrawal presented a formula to estimate the probability of inducing error [16]. In fact, because of the memoryness of sequential circuits and the dependence of execution sequence, the assumption of independent successive inputs is invalid. In our model, an exponentially distributed fault latency with rate α is assumed for simplicity. Since fault latency is generally much smaller than the life cycle, this assumption would not degenerate the accuracy of the whole model. Before an

error is induced by a fault, the system may transfer immediately into state D if signal level detection mechanisms cover this fault with probability one. Otherwise, the system enters state E. Another reason of direct transition from state F to state D is the execution of a diagnostic program. The transition duration from state F to state D is assumed to be exponentially distributed with parameter ω while the diagnositic program is running.

Once the system enters state E, the erroneous information starts to spread until function level detection mechanisms identify any unacceptable result. There are two paths to indicate this detection which have transition rates β and γ, respectively. β should be greater than γ since the existing fault in state E could induce more errors which may spread with high probability. In addition, the execution of a diagnostic program can also explore the fault in state E.

The model as described above is very general for covering the processes of error detection. The transition rates are dependent upon 1). the error detection mechanisms employed in the system, 2). the operations executed in the system, and 3). the characteristics of the concerned physical devices.

4. EVALUATION OF THE IMPACT OF ERROR LATENCY

4.1. Formulation of detection processes

Let a computer system incorporate the three types of error detection mechanisms discussed above. We are interested in both the useful computation time before the detection of error and the consequence of error latency. The diagnostic program is executed for period t_p after every normal operation

period t_n as shown in Figure 3. Thus the coverage of a single diagnostic, denoted as ξ, is equal to $1-e^{-\omega t_p}$ for each diagnostic period. The overhead for swapping between the normal operation and the diagnostic is denoted by t_v. The signal level detection provides a coverage c to detect error immediately. If the function level detection mechanism finds an error, the system may apply one of various recovery methods to rescue the contaminated message and computation. The recovery overhead is assumed to be a function of error latency, denoted as $R(t_{el})$ where t_{el} is error latency.

Since a latent fault is merely possible to harm system behavior we deal only with the error latency instead of the fault latency. Note that there is an absorbing state (Detection state) in our Markov model. To distinguish whether the error latency exists or not, we divide the state D into state D_1 and state D_2 where the transition to state D_1 has to go through state E, and state D_2 is reachable directly from state F if the fault is captured before the occurrence of error or an error is detected immediately when it occurs. For convenience let's number the states NF, F, FB, E, ENF, D_1, D_2 with i for i= 1,2..,7 and define the transition matrix $H_{7\times7}(t)$ as follow:

$$H_{7\times7}(t) = \begin{bmatrix} -\lambda & \frac{\lambda\nu}{\mu+\nu} & \frac{\lambda\mu}{\mu+\nu} & 0 & 0 & 0 & 0 \\ 0 & -(\mu+\alpha_1(t)+\alpha_2(t)) & \mu & \alpha_1(t) & 0 & 0 & \alpha_2(t) \\ 0 & \nu & -\nu & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -(\mu+\beta(t)) & \mu & \beta(t) & 0 \\ 0 & 0 & 0 & \nu & -(\nu+\gamma(t)) & \gamma(t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since the diagnostics are invoked periodically, transition rates $\alpha_1(t)$, $\alpha_2(t)$, $\beta(t)$, and $\gamma(t)$ are functions of time which are defined as follows:

$$\alpha_1(t) = \begin{cases} (1-c)\alpha & \text{if } n(t_n+t_p+t_v)<t\le n(t_n+t_p+t_v)+t_n \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_2(t) = \begin{cases} c\alpha & \text{if } n(t_n+t_p+t_v)<t\le n(t_n+t_p+t_v)+t_n \\ \omega & \text{otherwise} \end{cases}$$

$$\beta(t) = \begin{cases} \beta & \text{if } n(t_n+t_{p+t_v})<t\le n(t_n+t_p+t_v)+t_n \\ \omega & \text{otherwise} \end{cases}$$

$$\gamma(t) = \begin{cases} \gamma & \text{if } n(t_n+t_{p+t_v})<t\le n(t_n+t_p+t_v)+t_n \\ 0 & \text{otherwise} \end{cases}$$

Thus the transition probability matrix $P(t)=[p_{ij}(t)]$ can be solved by the forward Chapman-Kolmogorov equation [17]:

$$\frac{dP(t)}{dt} = P(t)H(t)$$

where $p_{ij}(t)$ is the probability that the system is in state j at time t given that the initial state is i. For the state probabilities, $\pi(t) = \left[\pi_1(t),\pi_2(t),\ldots,\pi_7(t)\right]$, we have the differential equation:

$$\frac{d\pi(t)}{dt} = \pi(t)\,H(t)$$

where $\pi_i(t)$ is the probability that the system is in state i at time t given $\pi(0)$. Because of the absorbing property of states D_1 and D_2, $\pi_6(\infty)+\pi_7(\infty)=1$.

4.2. Estimation of the probability of having an unreliable result

The execution of a task consists of parallel and/or serial execution of processes. We can always partition the task such that every process sends the computation result to its successors at the end of its execution and receives all the input data at the beginning of execution. Thus, each process can be con-

sidered as an atomic action [18]. Since an atomic action can be recovered very easily, we are not interested in the possible faults/errors within the atomic action if these faults/errors are detected. The more serious situation, namely the propagation of erroneous information through the system, occurs if the error can not be discovered by the end of execution. Let the probability that the system has at least one error at the output phase be p_e. Since the computation result may or may not be contaminated by the errors, we claim that p_e is also the probability of having an unreliable result.

Without periodic diagnostic, p_e can be represented easily by the error latency t_{el} and the process of error occurrence. Let $f_{t_{el}}(t)$ and $f_{er}(t)$ be the probability density functions of t_{el} and the time between two successive error occurrences induced by different types of fault. Then, the probability of having an unreliable result, p_e, is given by

$$p_e = \int_0^{T_k} f_{t_{el}}(t) \left[\frac{(1-c) \int_{T_k - t}^{T_k} f_{er}(\zeta) d\zeta}{1 - \int_0^{T_k - t} f_{er}(\zeta) d\zeta} \right] dt$$

where T_k is the process execution time. It is obvious that both the reduction of error latency and the avoidance of error can improve the final figure. Both density functions can be obtained from the forward Chapman-Kolmogorov equation which becomes homogeneous in this case.

When a scheduled periodic diagnostic is implemented for the process, the resultant p_e becomes a function of the time interval between the output moment and the previous diagnostic. The shorter this time interval, the more reliable the computation result. Because of the uncertainty of the task execution time, it is difficult to schedule a periodic diagnostic such that the system is tested just before the process moves into the output phase. Here, using the pro-

posed model, we evaluate the maximum p_e, denoted by $max(p_e)$, at which the time interval between task completion and the last diagnositic is t_n. For a process which sends out result at T_k, $max(p_e)$ is the probability that the system is in erroneous states (E or ENF) at time T_k, i.e. $max(p_e)=\pi_6(T_k)+\pi_7(T_k)$. In fact, because of the Markovian property in each transition, $max(p_e)$ is almost independent of the execution time T_k when T_k is much less than $\frac{1}{\lambda}$. The simulation results are graphed in Figures 4 and 5. In Figure 4, $max(p_e)$ starts to decrease only when each diagnostic has a higher coverage. Note that Tasar showed the running of diagnositc program for first $150\mu s$ can cover 98.46% faults of processor [9]. In Figure 5, we compare four different cases: 1). without diagnostic, 2). with periodic diagnostics and $c=0.6$, 3). with periodic diagnostics and $c=0.8$, and 4). with periodic diagnostics and doubling detection rate of function level detection mechanisms. It is noted that $max(p_e)$ is linearly related to the coverage of signal level detection and is changed exponentially with respect to the detection capability of function level detection mechanisms.

4.3. Calculation of computation loss

Given the characteristics of the signal and function level detection mechanisms which are incorporated in the system, a designer may question how much time is lost due to faults/errors and how much periodic diagnositics can improve the reliability and performance. Intuitively, periodic diagnostics can decrease the probability of having errors and can thus avoid the crash, but it certainly wastes the useful processing time. The following example is used to show the variations related to different parameters. If an error is detected after

execution interval T_d, the average computation loss due to fault and/or error, CL, is given by:

$$CL = \frac{(t_p+t_v)}{(t_n+t_p+t_v)} E(T_d) + (E(t_{el})+E(R(t_{el})))p_d$$

where p_d is the probability of detecting error by function level detection mechanisms, and $E(t_{el})$ is the mean value of error latency. $E(T_d)$ can be expressed as $p_d E(T_{d1})+(1-p_d)E(T_{d2})$ where T_{d1} and T_{d2} are the amount of time spent before the system is absorbed into states D_1 and D_2, respectively. T_{d1} and T_{d2} are random variables with pdf $\frac{p'_{16}(t)}{p_{16}(\infty)}$ and $\frac{p'_{17}(t)}{p_{17}(\infty)}$, respectively, where ' denotes the derivative with respect to time. The error latency is also a random variable with pdf $p'_{46}(t)$. Finally, the percentage of average computation loss is given by:

$$r = \frac{CL}{E(T_d)} = \frac{(t_p+t_v)}{(t_n+t_p+t_v)} \cdot p_{16}(\infty) + \frac{E(t_{el})+E(R(t_{el}))}{p_{16}(\infty)E(T_{d1})+p_{17}(\infty)E(T_{d2})}$$

The above equation indicates that the time wasted for executing periodic diagnostics is a dominating factor to the total computation loss when the system is highly reliable (i.e. the system has a small λ). However, only the task currently being executed suffers from the delay due to the error latency and recovery overhead. This delay in response may cause some serious damages to the system if execution of the task is time-critical. With $\lambda=10^{-6}$, $\alpha=0.02$, $\beta=0.2$, $\gamma=0.1$, $\omega=50$, the simulation results of the computation loss and the response delay due to error latency versus the period of a diagnostic cycle are ploted in Figure 6. Once the cost function of response time for a task and the recovery overhead are given, we can easily calculate the total loss and then decide the optimal diagnostic schedule which consists of the time interval between two successive diagnostics, t_n, and the coverage of each diagnostic, ξ.

Figure 7 presents the response delay due to error latency for different combinations of intermittent and permanent faults where greater error latency occurs when most faults have a short active time. The improvement in error latency by diagnostic appears notable only if the cycle time of diagnostic is not much greater than the fault's active time. However, the computation time is also wasted in this case. No ideal method so far has been established to diagnose the intermittent faults. Many computers are able to retry instructions whenever an error is detected. This method is useful to make the system survive intermittent faults, specially for reading or writing a tape or disk. Once an error can be detected immediately after its occurrence, the instruction retry method can also be applied in other parts of the system. This implies that signal level detection mechanisms should play an important role in fault-tolerant systems.

5. CONCLUSION

In this paper, we have presented two performance-related evaluations for fault-tolerant computers. These two are not usually included in the tranditional reliability models since such models do not deal with the process of error detection. The first evaluation, the probability of having an unreliable result, indicates the degree of confidence in computation result. The suspicion in the computation result is totally due to the deficiency of error detection process. Unfortunately, this deficiency can not be eliminated completely from any practical error detection mechanism. In the second evaluation, we take into account a more detailed computation loss resulting from the occurrence of error, its detection and the subsequent recovery. For many cases where a system

requires high overhead for error recovery or suffers from an erroneous output, the reliability analysis has to quantify these kinds of loss and has to provide a good method for estimating the total loss.

Though there are several assumptions to be justified through expriments, the model developed in this paper is general enough to include all aspects from fault occurrence to error detection and also various detection mechanisms. As shown in both evaluations, the model has systematically dealt with various aspects of error detection mechanisms. The results obtained here has high potential use in decision making during design or operation phase. The results also show favorable strategies of periodic diagnostics: 1). for time-critical tasks, one can derive an optimal diagnostic cycle to minimize the computation loss and the penalty of delayed response, 2). for noncritical tasks, the diagnostic is executed only when the system is idle.

ACKNOWLEDGMENT

The authors are grateful to Ricky Butler and Milton Holt at NASA Langley Research Center and C. M. Krishna at the University of Michigan for technical discussions and assistance.

REFERENCE

1. S. J. Bavuiso et al., "Latent Fault Modeling and Measurement Methodology for Application to Digital Flight Control", *Advanced Flight Control Symposium*, USAF Academy, 1981.

2. B. Courtois, "Some Results about the Efficiency of Simple Mechanisms for the Detection of Microcomputer Malfunction", *Proc. of the 9th Annual Int'l Symp. on Fault-Tolerant Computing*, 1979, pp. 71-74.

3. B. Courtois, "A Methodology for On-line Testing on Microprocessors", *Proc. of the 11th Annual Int'l Symp. on Fault-Tolerant Computing*, 1981, pp. 272-274.

4. J. J. Shedletsky, "A Rollback Interval for Networks with an Imperfect Self-Checking Property", *IEEE Trans. on Computers*, Vol. C-27, No. 6, June 1978, pp. 500-508.

5. H. Ball and F. Hardie, "Effects and Detection of Intermittent Failures in Digital Systems," *AFIP Conf. Proc.*, Fall 1969, pp. 229-235.

6. S. Osden, "The DC-9-80 Digital Flight Guidance System's Monitoring Techniques", *Proc. of the AIAA Guidance and Control Conf.*, 1979, pp. 64-79.

7. N. L. Gunther and W. C. Carter, "Remarks on the Probability of Detecting Faults", *Proc. of th 10th Annual Int'l Symp. on Fault-Tolerant Computing*, 1980, pp. 213-215.

8. J. J. Shedletsky, "Random Testing: Practicality vs. Verified Effectiveness", *Proc. of the 7th Annual Int'l Symp. on Fault-Tolerant Computing*, 1977, pp. 175-179.

9. V. Tasar, "Analysis of Fault-Detection Coverage of a Self-Test Software Program", *Proc. of the 8th Annual Int'l Symp. on Fault-Tolerant Computing*, 1978, pp. 65-74.

10. M. T. Breuer, "Testing for Intermittent Faults in Digital Circuits," *IEEE Trans. on Computers*, Vol. C-22, No. 3, March 1973, pp. 241-246.

11. I. Koren and S. Y. H. Su,"Reliability Analysis of N-modular Redundancy Systems with Intermittent and Permanent Faults", *IEEE Trans. on Computers*, Vol. C-28, No. 7, July 1979, pp. 514-520.

12. Y. K. Malaiya and S. Y. H. Su, "Reliability Measure of Hardware Redundancy Fault-Tolerant Digital Systems with Intermittent Faults", *IEEE Trans. on Computers*, Vol. C-30, No. 8, August 1981, pp. 600-604.

13. Y. W. Ng and A. A. Avizienis, "A Unified Reliability Model for Fault-Tolerant Computers," *IEEE Trans. on Computers*, Vol. C-29, No. 11, Nov. 1980, pp. 1002-1011.

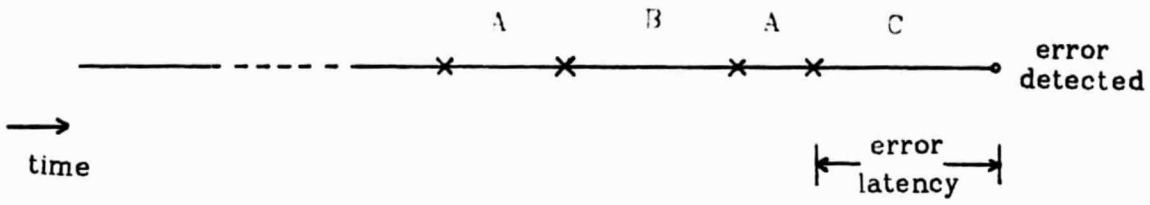
14. T. H. Lada and A. L. Hopkins, Jr., "Survival and Dispatch Probability Models of the FTMP," *Proc. of the 8th Annual Int'l Symp. on Fault-Tolerant Computing*, 1978, pp. 37-43.

15. K. S. Trivedi and R. M. Geist, "A Tutorial on the CARE III Approach to Reliability Modeling", *NASA Report*, No. 3488, 1981.

16. V. D. Agrawal, "An Information Theoretic Approach to Digital Fault Testing", *IEEE Trans. on Computers*, Vol. C-30, No. 8, August 1981, pp. 582-587.

17. L. Kleinrock, *Queueing Systems, Volume 1: Theory*, John Wiley & Sons Inc, 1975.

18. D. B. Lomet, "Process Structing, Synchronization, and Recovery Using Atomic Actions," *Proc. ACM Conf. Language Design for Reliable Software*, SIGPLAN Notices 12, no. 3, March 1977, pp. 128-137.



Duration A: A fault exists and is active in the system.
Duration B: The fault becomes inactive.
Duration C: Errors exist in the system.

Figure 1. The error detection process.

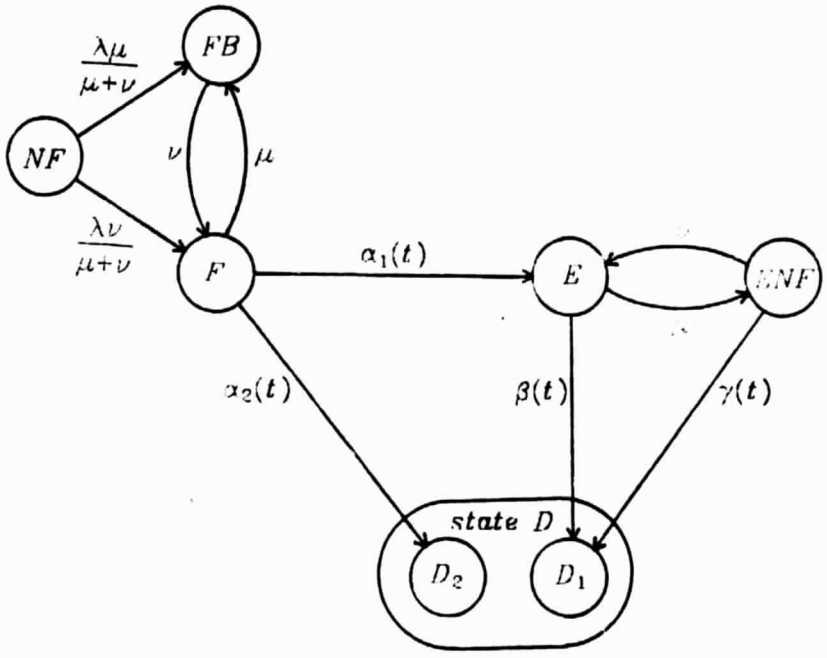


Figure 2. The model for error detection process.

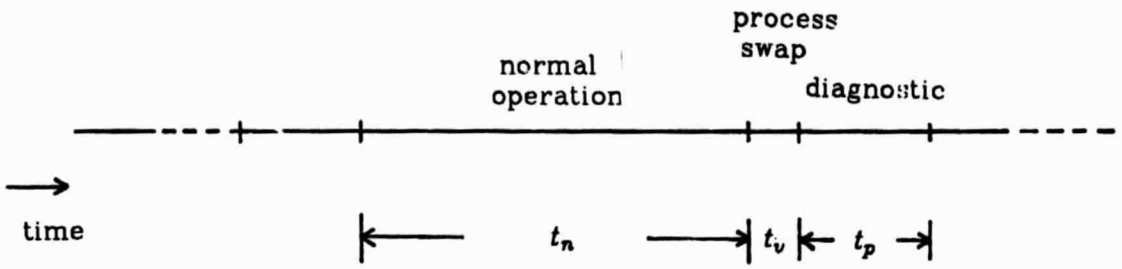


Figure 3. A cycle of periodic diagnostic.

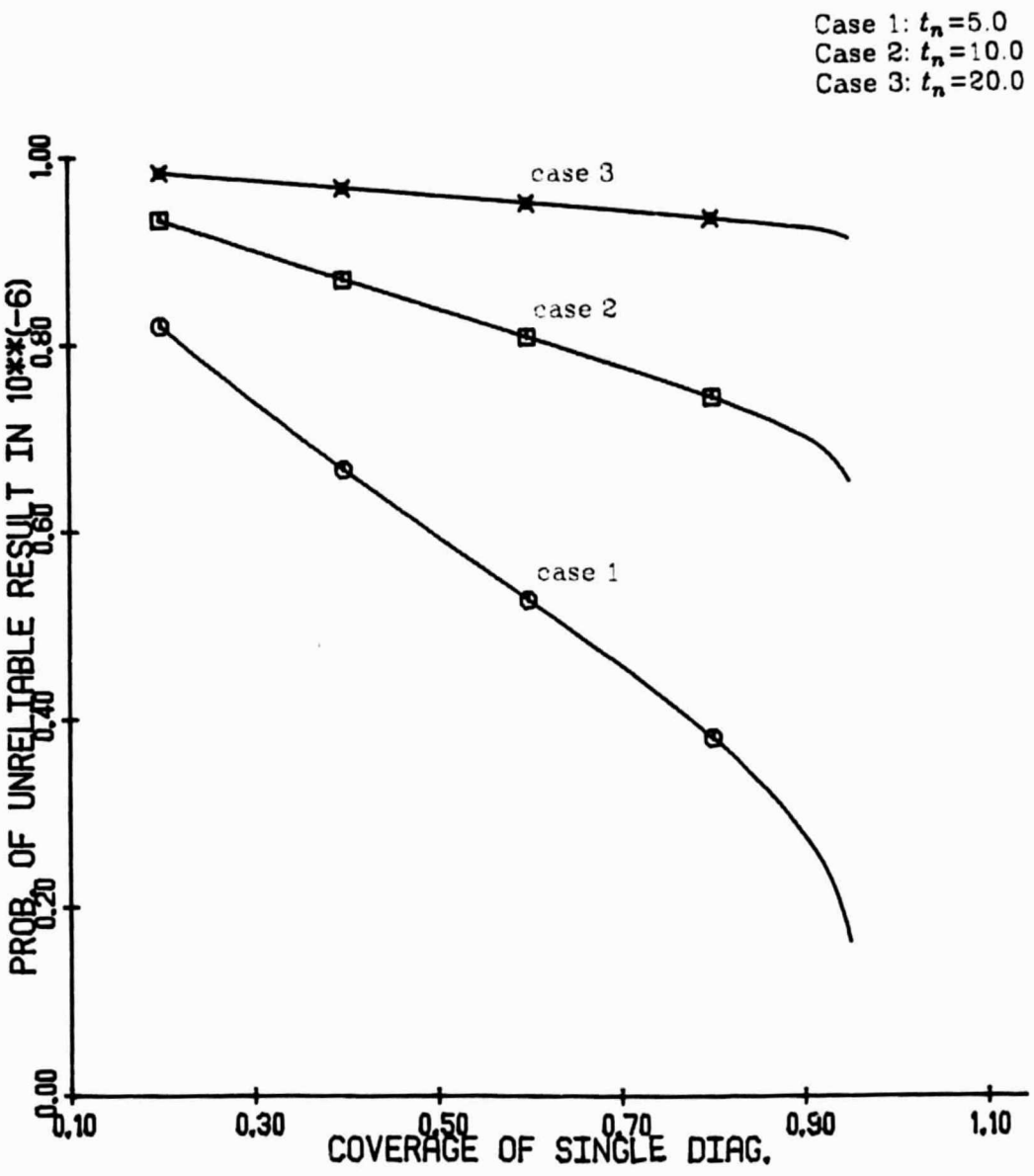


Figure 4. Probability of having an unreliable result versus coverage of single diagnostic.
($\lambda=10^{-6}$, $\mu=0.1$, $\nu=0.2$, $\alpha=0.2$, $\beta=0.5$, $\gamma=0.1$, $c=0.6$)

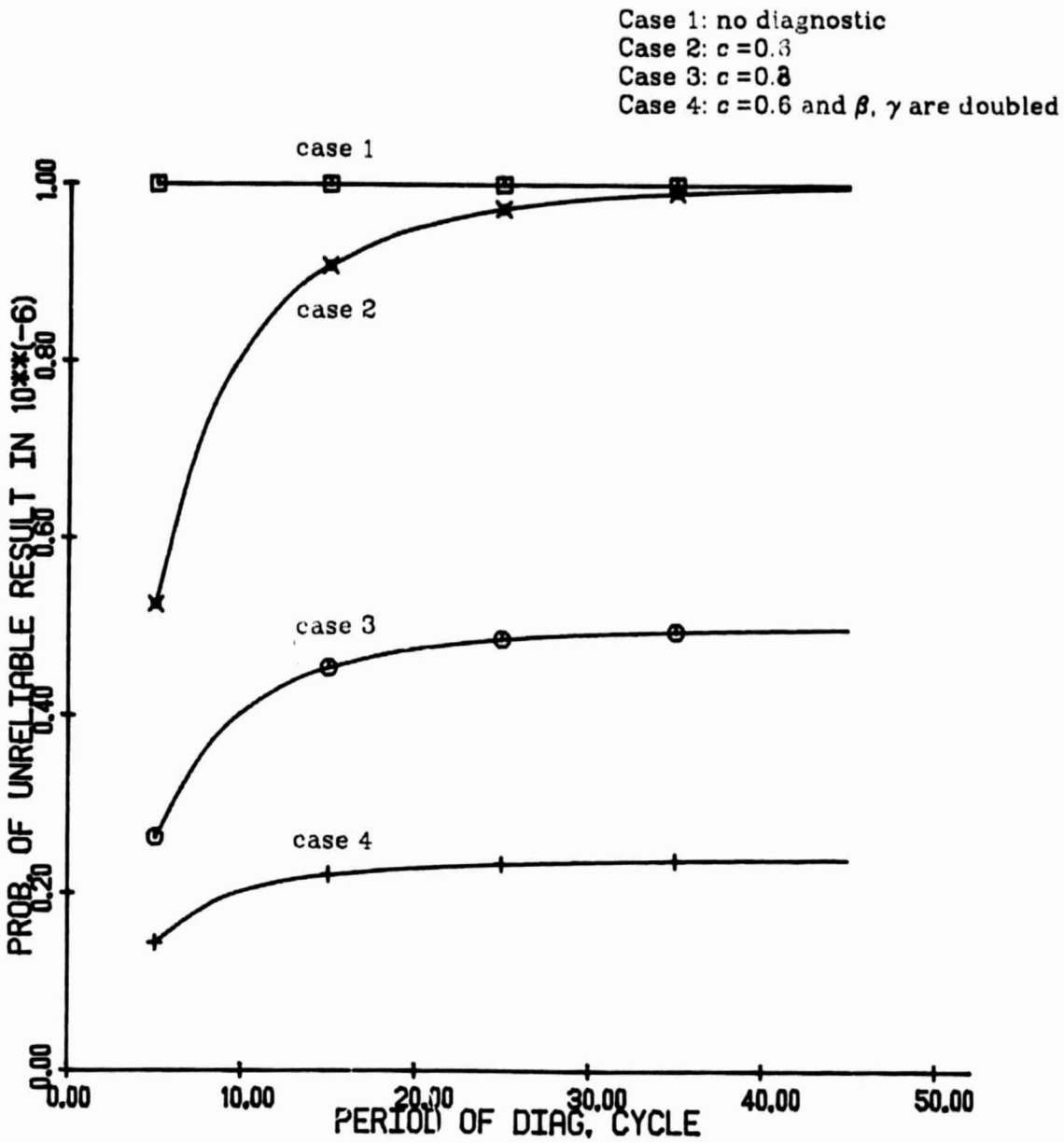


Figure 5. Probability of having an unreliable result versus period of diagnostic cycle.
($\lambda=10^{-6}$, $\mu=0.1$, $\nu=0.2$, $\alpha=0.2$, $\beta=0.5$, $\gamma=0.1$, $\xi=0.6$)

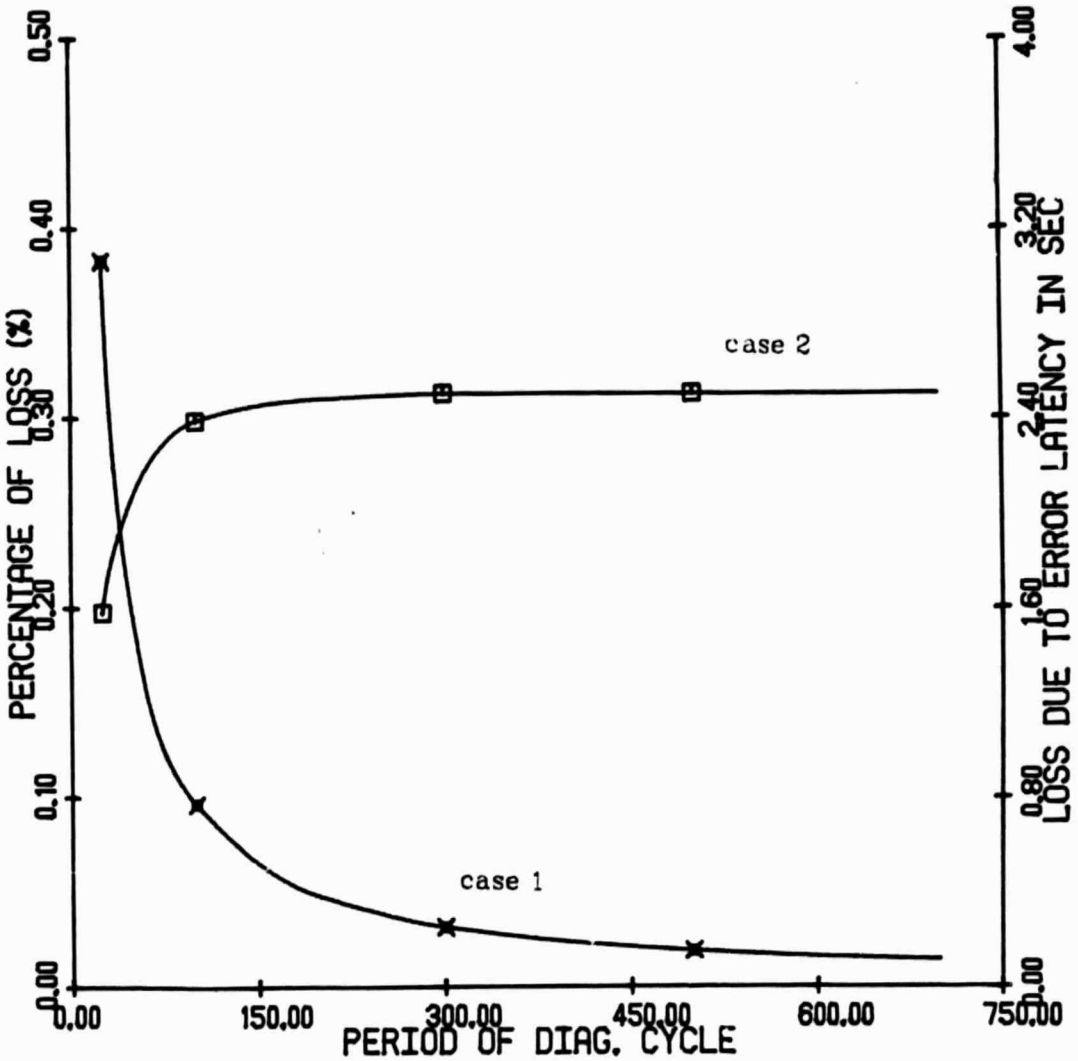


Figure 6. Percentage of total loss (case 1) and average loss due to error latency (case 2) versus period of diagnostic cycle. ($\lambda=10^{-6}$, $\mu=\nu=0.2$, $\alpha=0.05$, $\beta=0.2$, $\gamma=0.1$, $\omega=50.0$, $\xi=0.9$, $c=0.6$)

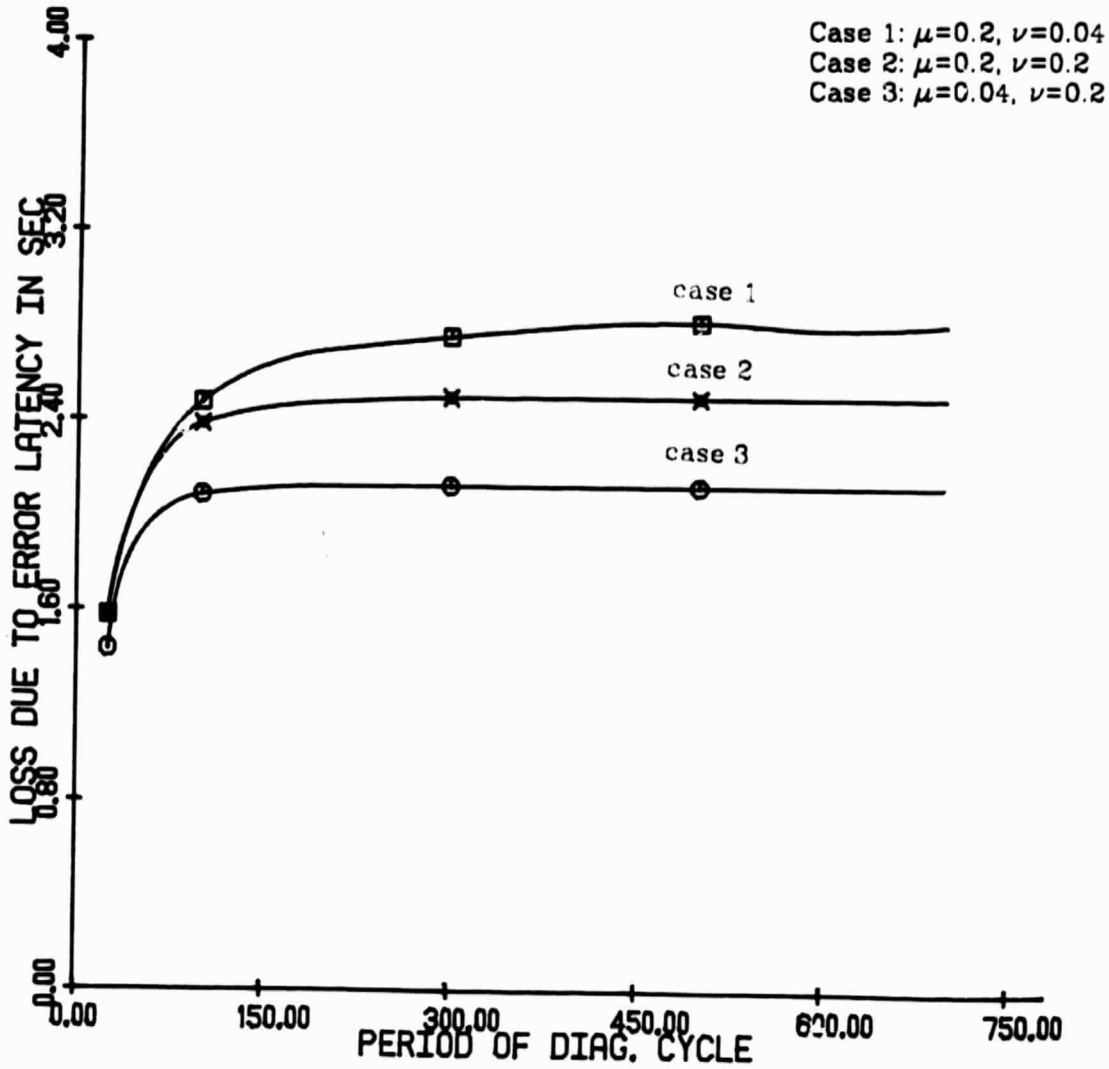


Figure 7. Average loss due to error latency versus period of diagnostic cycle.
($\lambda=10^{-6}$, $\alpha=0.05$, $\beta=0.2$, $\gamma=0.1$, $\omega=50.0$, $\xi=0.9$, $c=0.6$)