STUDIES ON CONFIGURATION AND RECOVERY TECHNIQUES FOR FAULT-TOLERANT COMPUTING SYSTEMS

January 1992

Satoshi Fukumoto

STUDIES ON CONFIGURATION AND RECOVERY TECHNIQUES FOR FAULT-TOLERANT COMPUTING SYSTEMS

Satoshi Fukumoto

January 1992

STUDIES ON CONFIGURATION AND RECOVERY TECHNIQUES FOR FAULT-TOLERANT COMPUTING SYSTEMS

Satoshi Fukumoto

January 1992

Dissertation submitted to the Faculty of Systems Engineering, Doctoral Course of the Graduate School of Engineering, Hiroshima University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Engineering).

ABSTRACT

It is of great importance to operate a computer system with high reliability. Several techniques to achieve the high reliability of a computer system have been proposed and implemented in the real computer systems. This dissertation discusses configuration and recovery techniques for fault-tolerant computing systems, for which stochastic models are presented to evaluate performance and/or reliability. Chapter 1 gives introduction for configuration and recovery techniques based on the concept of *redundancy*. Chapter 2 presents two models for evaluating database recovery mechanisms. The first model discusses the recovery mechanism with periodical checkpoint generations. The second model further discusses the recovery mechanism in the situation where the road of the system varies with time in a shape of a cycle. Chapter 3 presents a model for evaluating the improvement on system reliability by retries based on time redundancy. In Chapter 4, two models for multi-processor systems are proposed from the viewpoint of transaction assignment, and are compared using the reliability/performance measures. Chapter 5 discusses a reliability evaluation software package tool for a system formulated by a continuous-time Markov chain with many states. Finally, Chapter 6 summarizes the results obtained in the dissertation, and discusses the further research works on configuration and recovery techniques.

ACKNOWLEDGEMENTS

The author would like to express his thanks to Professor Shunji Osaki for his continuing support, encouragement and guidance for this dissertation.

The author wishes to thank the members of the dissertation reviewing committee: Professor Noriyoshi Yoshida, Professor Mitsuo Nagamachi, and Professor Masatoshi Sakawa for their careful review of this dissertation.

The author is also grateful to Associate Professor Shigeru Yamada and Research Associate Dr. Hiroaki Tanaka of Hiroshima University for their continuing encouragement and useful discussions. The author has also received invaluable cooperation and suggestions from many individuals during the course of this work. In particular, the author wishes to thank Associate Professor Naoto Kaio of Hiroshima Shudo University, Mr. Hideshi Ohshimo of Nishikawa Rubber Co., Ltd., and Associate Professor Kazumi Yasui of Aichi Institute of Technology.

Finally, it is a special pleasure to acknowledge the hospitality and encouragement of the past and present members of Professor Osaki's Laboratory.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Configuration and Recovery Techniques for Fault-Tolerance

In our daily lives we are usually using computer systems to keep our high quality lives and highly social activities. Such computer systems are telephone exchange systems, communication systems, banking systems, train and/or airline seat reservation systems, and so on. Demands of applying such computer systems has a remarkably increasing tendency following the progress of computer hardware and software technologies. Without such computer systems, we cannot keep our high quality lives.

Once failure of such computer systems takes place, it might be costly, dangerous, and might cause confusion in our society. Even now, the serious crisis is facing us. To achieve the high reliability of such computer systems, we should propose and implement the highly reliability techniques for computer systems from the viewpoint of hardware and software. The main objective of this dissertation is to introduce and evaluate such high reliability and/or performance techniques for computer systems.

We have two approaches for improving the reliability of the system, which are called *fault-avoidance* and *fault-tolerance* [1, 2]. Fault-avoidance is a technique which decreases the possibility of fault occurrence. Adoption of the high-reliable components is a typical example for such a technique. On the other hand, fault-tolerance is a technique by which the system tries to tolerate the faults, considering that a fault can not be completely prevented. There are three principal stages in fault-tolerant techniques [1, 2]: error detection and correction, configuration and recovery, and, diagnosis and repair.

This dissertation concentrates our interest on configuration and recovery techniques in fault-tolerant computing systems [11, 12], for which we introduce stochastic models to evaluate performance and/or reliability [13-20]. It is of great importance for system design, operation and maintenance, to evaluate the reliability and/or performance qualitatively and/or quantitatively.

A system configuration technique and a recovery technique have a closely mutual relation, and are realized by the concept of *redundancy*. We can classify redundancies into three types from the viewpoint of their actualization methods [1, 2] as follows:

Hardware redundancy: The system has additional redundant hardware modules or redundant information.

Software redundancy: The system has additional redundant software modules including the software which controls hardware redundancy and time redundancy.

Time redundancy: The system spends redundant time.

We can also classify redundancies into two types from the viewpoint of their principles [1, 2] as follows:

Masking redundancy: Faults are masked by fixed redundant configurations, and are not recognized by the outside. This redundancy is also called static redundancy.

Dynamic redundancy: Errors caused by faults are detected, and the recovery procedures are executed.

A classification of the typical redundant techniques is shown in Table 1.1. Redundancy by error correcting codes is the first-step of fault masking and, is also called an information redundant technique [1, 2]. TMR (Triple Modular Redundancy) has three equivalent hardware modules and decides the output by a majority vote of them [1, 2, 19]. On the other hand, standby redundancy and graceful degradation are the techniques which execute the reconfiguration of system components by redundant hardware modules. In standby redundancy, redundant modules are used for replacement of failed modules [1, 19]. Graceful degradation uses even the redundant modules in normal operation, and separates failed modules on the occurrence of the failure [19, 21, 22]. Similarly, N-version programming and recovery blocks have multi-version software modules, and use the redundancy for a majority vote and replacement, respectively [1, 13]. The rollback is a fundamental procedure for a recovery action [1, 2]. If a failure spoils the the system states, the consistent states which are called *checkpoints* and have been collected in a safe place at prespecified time points, are reintroduced to the system [23-25]. The retry is a technique which attempts repeatedly the same action interrupted by a failure to recover from a transient or intermittent failure [26, 27].

Based on the redundancies above, when a fault takes place, the system carries out the recovery procedures as shown in Fig. 1.1 [1, 2] so as to restore the correct states and restart the normal operation.

The dissertation studies the following four themes concerned with the above configuration and recovery techniques; Database Recovery, Retry procedure, Multi-processor systems (Gracefully degrading systems), Software package tool for reliability evaluation.

Table 1.1: A classification of typical redundant techniques.

	Masking Redundancy	Dynamic Redundancy
Hardware Redundancy	Error Correcting Code TMR(Triple Modular Redundancy)	Stand by Redundancy Graceful Degradation
Software Redundancy	*N*-Version Programming	Recovery Blocks Rollback
Time Redundancy		Retry

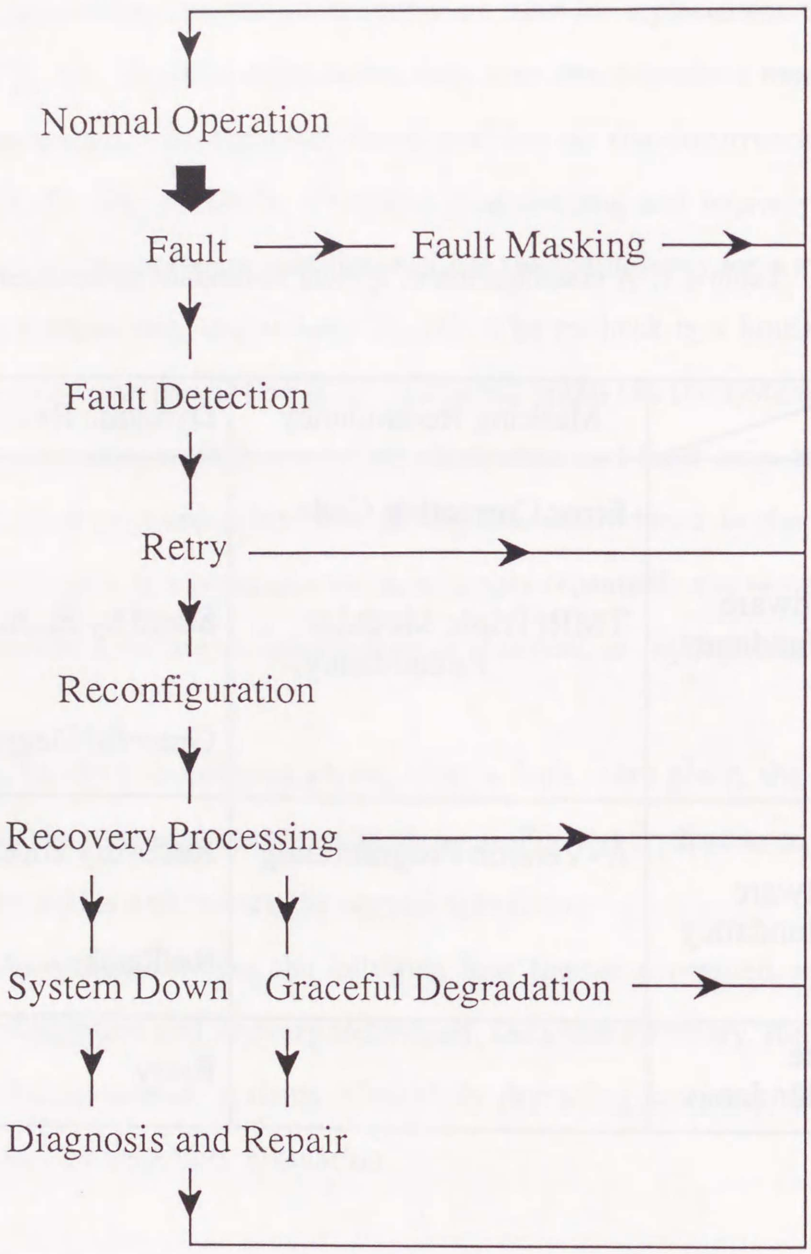

Figure 1.1: Recovery procedures for a system.

1.2 Organization of Dissertation

In this section, we summarize the dissertation. The dissertation is organized by Introduction, Chapter 2-5, and Conclusion.

Chapter 2 discusses database recovery techniques [23, 24]. Two models are presented for evaluating the recovery mechanisms. The first model evaluates the recovery mechanism with periodical checkpoint generations [24]. The expected recovery time and the availability for one cycle are derived under the assumption of an arbitrary failure-time distribution. In particular, we analytically obtain the optimum checkpoint interval with the maximum availability in the case of an exponential distribution. We numerically calculate the above results by assuming Weibull distributions. We further discuss the numerical results in varying the parameters that we define in our model, and show the impact of these parameters on the recovery mechanism. The second model discusses the recovery mechanism with checkpoint generations in a varying load situation. The density of checkpoint generations is analytically derived from minimizing the expected total overhead to completion of a phase, and this density yields the optimum sequence of checkpoint generations measured in unit of update pages. We further present the numerical examples for the results obtained and show that the sequence gives effective checkpoint generations.

Chapter 3 presents a model for evaluating the improvement on reliability by retries based on time redundancy [1, 2, 26]. Taking account of the behavior of intermittent and permanent failures [27], we consider the evaluation model which manages the system maintenance with the prescribed number of successful retries. Analysis of the model, by applying the Markov renewal processes [28], yields the availability and the mean time between failures in the steady-state. We further describe a calculation method for the availability in the transient-state introducing continuous time Markov chains and a randomization technique [29, 30]. Numerical illustrations for the results above shows the several important properties of retry procedures.

Chapter 4 discusses a multi-processor system which is one of the typical fault-tolerant computing systems, and is also called *A Gracefully Degrading System* from its redundant technique [1, 2, 19]. The system is assumed to be composed of two processors and buffer(s), and is evaluated taking account of the reliability, performance and computational demands simultaneously. We propose two models for the system from the viewpoint of transaction assignment. Applying Markov renewal [28] and queuing theories [31], we obtain the reliability/performance measures for each model. Using the numerical results of our models, we compare two models and show the impact of transaction assignment on the evaluation measures based on our numerical examples.

In Chapter 5, we discuss a reliability evaluation software package tool for a system formulated by a continuous-time Markov chain with many states.

The randomization technique [29, 30] is discussed to derive the transient solution for the Markov chain. The software package tool is implemented by using the randomization technique and introducing a new idea of identifying when the transient solution converges to the steady-state solution in advance. Numerical examples are illustrated by using our software package tool to evaluate the optimal maintenance policies for computing systems. Some interesting maintenance policies for compute systems are suggested from the numerical examples.

The conclusion summarizes the results obtained in the dissertation. Finally, discussion of the future research works on configuration and recovery techniques concludes the dissertation.

References

[1] D. P. Siewiorek and R. S. Swarz (eds.): *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, Massachusetts (1982).

[2] M. Mukaidono (ed.): *Introduction to Highly Reliable Techniques for Computer Systems*, Japanese Standards Association, Tokyo (1988) (in Japanese).

[3] Reliability Center for Electronic Components of Japan: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 57th Fiscal Year of Showa*, Reliability Center

for Electronic Components, Tokyo (1983) (in Japanese).

[4] Japan Information Processing Development Center: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 58th Fiscal Year of Showa*, Japan Information Processing Development Center (1984) (in Japanese).

[5] Japan Information Processing Development Center: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 59th Fiscal Year of Showa*, Japan Information Processing Development Center, Tokyo (1985) (in Japanese).

[6] Japanese Standards Association: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 60th Fiscal Year of Showa*, Japanese Standards Association, Tokyo (1986) (in Japanese).

[7] Japanese Standards Association: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 61th Fiscal Year of Showa*, Japanese Standards Association, Tokyo (1987) (in Japanese).

[8] Japanese Standards Association: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems)*

for the 62th Fiscal Year of Showa, Japanese Standards Association, Tokyo (1988) (in Japanese).

[9] Japanese Standards Association: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the 63th Fiscal Year of Showa*, Japanese Standards Association, Tokyo (1989) (in Japanese).

[10] Japanese Standards Association: *The Report of Research Studies on Highly Reliable System Techniques (Electronic Application Systems) for the First Fiscal Year of Heisei*, Japanese Standards Association, Tokyo (1990) (in Japanese).

[11] A. Avižienis: "Fault-Tolerant System", *IEEE Trans. Comput.*, Vol. C-25, No. 12, pp. 1304-1311 (1976).

[12] A. Avižienis and J. C. Laprie: "Dependable computing: From concepts to design diversity", *Proc. IEEE*, Vol. 74, No. 5, pp. 629-638 (1988).

[13] A. Avižienis, H. Kopetz and J. C. Laprie (eds.): *The Evolution of Fault-Tolerant Computing*, Springer-Verlag, Wien (1988).

[14] A. Costes, C. Landrault and J. C. Laprie: "Reliability and Availability Models for Maintained Systems Featuring Hardware Failure and Design Faults", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 548-560 (1978).

[15] Y. W. Ng and A. Avižienis: "A Unified Reliability Model for Fault-Tolerant Computers", *IEEE Trans. Comput.*, Vol C-29, No. 11, pp. 1002-1011 (1980).

[16] S. Osaki and T. Nishio: *Reliability Evaluation of Some Fault-Tolerant Computer Architectures*, Springer-Verlag, Berlin (1980).

[17] M. D. Beaudry: "Performance-Related Reliability Measures for Computing Systems", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 540-547 (1978).

[18] J. F. Meyer: "On Evaluating the Performability of Degradable Computing Systems", *IEEE Trans. Comput.*, Vol. C-29, No. 8, pp. 720-731 (1980).

[19] S. Osaki: "Performance/Reliability Measures for Fault-Tolerant Computing Systems", *IEEE Trans. Reliab.*, Vol. R-33, No. 4, pp. 268-271 (1984).

[20] M. Nakamura and S. Osaki: "Performance/Reliability Evaluation of a Multi-Processor System with Computational Demands," *Int. J. System Sci.*, Vol. 15, No. 1, pp. 95-105 (1984).

[21] F. A. Gay and M. L. Ketelsen: "Performance Evaluation for Gracefully Degrading Systems", in *Proc. 9th FTCS*, Madison, Wisconsin, pp. 51-58 (1979).

[22] T. Nakagawa and S. Osaki: "Stochastic Behavior of a Two-Unit Standby Redundant System", *INFOR*, Vol. 12, No. 1, pp. 66-77 (1974).

[23] J. M. Verhofstadt: "Recovery Techniques for Database Systems", *ACM Comput. Surv.*, Vol. 10, No. 2, pp. 167-196 (1978).

[24] T. Haerder and A. Reuter: "Principles of Transaction-Oriented Database Recovery", *ACM Comput. Surv.*, Vol. 15, No. 4, pp. 287-317 (1983).

[25] R. Koo and S. Toueg: "Checkpointing and Rollback-Recovery for Distributed Systems", *IEEE Trans. Softw. Eng.*, Vol. SE-13, No. 1, pp. 23-31 (1987).

[26] H. Inose (ed.): *Reliable Computer Systems*, IPS Japan, Tokyo (1977) (in Japanese).

[27] Y. K. Malaiya: "Linearly Correlated Intermittent Failures", *IEEE Trans. Reliab.*, Vol. R-31, No. 2, pp. 211-215 (1982).

[28] T. Nakagawa and S. Osaki: "Markov Renewal Processes with Some Non-Regeneration Points and Their Applications to Reliability Theory", *Microelectron. Reliab.*, Vol. 15, pp. 633-636 (1976).

[29] D. Gross and D. R. Miller: "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes", *Oper. Res.*, Vol. 32, No. 2, pp. 343-361 (1984).

[30] E. D. S. Silva and H. R. Gail: "Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization", *J. ACM*, Vol. 36, No. 1, pp. 171-193 (1989).

[31] L. Kleinlock: *Queueing Systems; Volume I: Theory*, John Wiley and Sons, New York (1975).

Chapter 2

Evaluation for Database Recovery Mechanisms

This chapter discusses database recovery techniques. Two models are presented for evaluating the recovery mechanisms. First, we describe a model in order to treat the changing of a failure rate of the system with time. Secondly, we develop a model in the situation where the road of the system varies with time in a shape of a cycle.

2.1 A Recovery Mechanism with Periodical Checkpoint Generations

2.1.1 Introduction

It is of great importance to recover a computer system after a system failure in order to restart the system operation. Such a recovery action is one of the fault-tolerant technologies, and must be considered in operation of all systems. However, it is particularly important in database systems which support on-line transaction processing (OLTP) system such as banking and airline seat reservation systems. Therefore, several recovery techniques are applied to the database management system (DBMS) that controls the database recovery actions. Verhofstadt [1] surveyed the standard database recovery mechanisms. Haerder and Reuter [2] introduced a framework for establishing and evaluating the basic concepts for fault-tolerant database operation. In this section, we discuss evaluation for a database recovery action.

In general, checkpoints are used to limit the amount of the data processing for the recovery action. Generating a checkpoint means that the results of a complete transaction(s) in the database buffer are collected in a safe place. When a system failure occurs, we trace the transaction processing back to the checkpoint instead of the starting point of the system operation, and reconstruct the logical consistency of the database which had been constructed just before the failure.

It is very important to decide the optimum checkpoint time interval, if periodical checkpoint generations are used, considering the influence of the overhead during normal operations on the system performance. If we make the interval too short, we have to incur the high overhead time for frequent checkpoint generations, and, conversely, if we make it too long, we have to spend much recovery time when the system failure occurs. Chandy et al. [3], and Gelenbe [4] discussed the optimum checkpoint interval, by assuming that the failures occur as a Poisson process. Sumita et al. [5] discussed such problems by assuming a nonhomogeneous Poisson process. In these previous works, one cycle for evaluating the recovery action is the interval between two successive checkpoint generations, and therefore, the completion of the checkpoint generation implies a renewal of the system in terms of a failure rate. Since the failure rate of the system seems to be unchanged by the checkpoint generation, it is necessary to re-examine the situation above. Young [6] derived the optimum checkpoint interval for the endless job processing, minimizing the time lost caused by the checkpoint generations and recovery from the failure due to the exponential distribution. Reuter [7] proposed a set of models that evaluate mean value of transaction rate for comparing recovery techniques of the taxonomy in the reference [2]. In these works, unlike the above assumption, one cycle is the interval from the beginning of the processing to the completion of the recovery. However, we can not use the models in case where the failure rate of the system changes with time, e.g., the system is in the early phase or wearout phase.

We discuss the study of the above problem, proposing a new model to evaluate the database recovery action. In our model, one cycle is defined as the interval from the start of the system operation to the completion of the recovery action after the failure. We are especially interested in evaluation for the recovery actions, under the assumption of an arbitrary failure-time distribution [8]. This is of great use for various kinds of distributions, such as decreasing failure rate (DFR) distributions in the early stage of system operation. In the next subsection, we discuss general database recovery mechanisms and checkpoint generations to explain the environment of the recovery action which we evaluate in this paper. In Subsect. 2.1.3, we describe our new model based on the concepts above, introducing several assumptions. In order to evaluate the recovery action, we obtain the expected recovery time and the availability for one cycle as a function of checkpoint interval, in Subsect. 2.1.4. We further derive analytically a formula of the optimum checkpoint interval with the maximum availability, in a case where the failure-time distribution obeys an exponential distribution, generalizing the formula obtained by Young [6]. Finally, we numerically evaluate the recovery actions and the optimum checkpoint intervals from the above results by assuming Weibull distributions including a DFR distribution in Subsect. 2.1.5. We discuss the numerical results in varying the parameters such as the mean time to the failure, the overhead for a checkpoint generation, etc., and show the impact of these parameters on the recovery action.

2.1.2 Recovery Mechanisms and Checkpoints

A design goal of the recovery mechanisms is to be able to reconstruct the logical consistency of the database on the occasion of a system failure. This fact implies that once we have allowed a transaction to commit its results to the database, we must protect the effects of the transaction from any system failure, conversely, if we have not allowed a transaction to commit yet, we must not leave the incomplete effects of the transaction caused by the system failure. In the former case, we call the transaction *a complete transaction*, and in the latter case, *an incomplete transaction*.

In general, an update transaction modifies the pages in the database buffer as required, and the modified pages will be propagated to the secondary (nonvolatile) storage of the database according to the buffer management. When a system failure occurs, it is assumed that the database buffer loses the information on modified pages, since the system failure forces the system to terminate the transaction processing in an uncontrolled manner. Therefore, we must assume that the contents of the secondary storage are not enough to satisfy the logical consistency of the database.

The recovery action in such a situation consists of two operations. One is UNDO operation, which rolls back the effects of all incomplete transactions from the database, and the other is REDO operation, which reflects the results of all complete transactions in the database. The amount of UNDO operation depends on the number of pages which have been propagated to the secondary storage before the failure even though the pages have been modified by incomplete transactions. On the other hand, the amount of REDO operation depends on the number of pages which were remaining in the buffer at the time of the failure even though the pages have been modified by complete transactions. These are regulated by the buffer manager according to the capacity of the database buffer. In order to process these operations, redundant information for each transaction is collected in a log file which survives system failures. The log file contains the following information: BOT (begin of transaction) records; information for UNDO operation; information for REDO operation; and EOT (end of transaction) records. Applying the information of the log file to the contents of the secondary storage, the DBMS enables us to reconstruct the database to be logically consistent, and restart processing normally.

However, as a worst-case assumption, i.e., if there is no propagation to the secondary storage before the system failure, we have to process REDO operation from the beginning of the log file. To prevent such heavy overhead, an additional redundant method should be taken. Hence we institute the specific provisions which are called *checkpoints*, and in which the results of a complete transaction(s) are collected in a safe place such as the secondary storage or the log file. Checkpoints are used to define and limit the amount of REDO operation.

In particular, if checkpoint generation means that the pages modified by a transaction are propagated to the secondary storage before the transaction is recognized as a complete transaction, no REDO operation is required in the recovery action after the system failure. Such a type of checkpoints is called *Transaction-Oriented Checkpoints* (*TOC*) in the classification presented by Haerder and Reuter [2]. In this approach, however, every time an update transaction is recognized as a complete transaction, the pages which are modified by that transaction and remaining in the buffer must be propagated to the secondary storage. This regulation is disadvantageous to the propagation overhead for what we call *hot spot* pages which remain in the buffer for a long time since many transactions modify these pages again and again frequently. Although *TOC* is excellent in terms of REDO operation, its overhead during normal processing is too high to be used in large applications, since the number of hot spot pages increases in proportion to the buffer capacity.

In this section, we discuss the periodical checkpoint generations, in which all modified pages in the buffer are reflected in secondary storage, in other words, contents of the secondary storage are synchronized with the database buffer. After the system failure, we process REDO operation from the most recent checkpoint. In this case, we can limit the extent of REDO operation to the checkpoint interval, that is, the time interval between two successive checkpoint generations. Such a type of checkpoints corresponds to *Transaction-Consistent Checkpoints* (*TCC*) or *Action-Consistent Checkpoints* (*ACC*) in the above classification[2]. *TOC* and *fuzzy checkpoints*, which have the advantage of low overhead during normal operations [2], are of great interest, but we do not discuss these types.

2.1.3 Model for a Recovery Mechanism

In this section, all results of complete transactions remaining in the database buffer are reflected to the secondary storage periodically with the checkpoint interval T. We spend the overhead time C for a checkpoint generation, which is independent of the checkpoint interval T. This assumption might seem not to apply in practice, e.g., in a case where T is a short period and the number of the modified pages are small. However, in large applications which are the subjects of our model mainly, the assumption is justified by the following reasons:

- After the (re)start of the normal operation, the amount of update information in the buffer reaches to an upper bound of the capacity within a very short term. Then, that is controlled to be constant by the buffer manager propagating the modified pages other than hot spot ones. We can neglect the above transient term compared with the interval T.
- The hot spot pages remaining in the buffer for a long time, contain many updates which increase with the interval T. On the other hand, the number of these pages is almost constant, that is, the propagation overhead for the pages are constant independent of the interval T.

We assume that the time to the failure X obeys an arbitrary distribution $F(t)$ with a finite mean $E[X]$. We define the survival probability $\overline{F}(t) \equiv$

$1 - F(t)$.

Let $R(x)$ be the recovery function, which denotes the required time interval for the recovery actions after the failure for x units of time for normal operations. Hence, we have to spend the recovery time $R[t - (k-1)(T+C)]$ when the system failure takes place at time t which is in kth normal operation. If the system failure occurs during checkpoint generations, the results of transactions processed in time interval T are lost and we have to spend the recovery time $R(T)$. The concrete recovery function is introduced in the later.

We define a cycle as the interval from the start of the system operation to the completion of the recovery action after the failure. A sample function of our model for one cycle, is depicted in Fig. 2.1.

2.1.4 Analysis

Let us derive the expected recovery time and the availability for one cycle.

In the case where the failure occurs in the kth normal operation, the total time for the normal operations, Y, is given by $Y = X - (k-1)C$. In the case where the failure occurs in the kth checkpoint generation, the total time Y is given by $Y = kT$. Let $E[Y \mid t_1 \leq X < t_2]$ be the conditional expectation of Y given $t_1 \leq X < t_2$, and $T_k = k(T+C)$. We obtain U_T, the

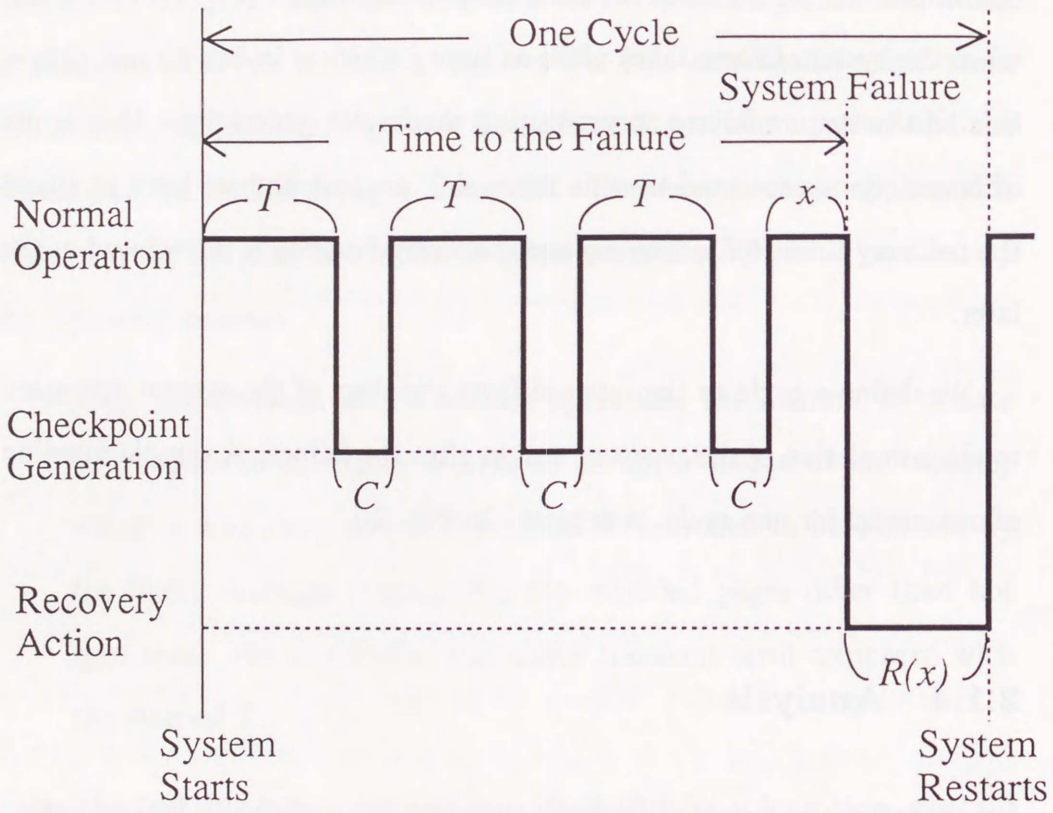

Figure 2.1: A sample function of the model for one cycle.

expected total time for the normal operations before the failure:

$$
\begin{aligned}
U_T &= \sum_{k=1}^{\infty}\Big[E[Y \mid T_{k-1} \le X < T_{k-1}+T]\cdot \Pr\{T_{k-1} \le X < T_{k-1}+T\} \\
&\qquad + E[Y \mid T_{k-1}+T \le X < T_k]\cdot \Pr\{T_{k-1}+T \le X < T_k\}\Big] \\
&= \sum_{k=1}^{\infty}\Bigg[\frac{\int_{T_{k-1}}^{T_{k-1}+T}[t-(k-1)C]\,dF(t)}{\Pr\{T_{k-1} \le X < T_{k-1}+T\}}\cdot \Pr\{T_{k-1} \le X < T_{k-1}+T\} \\
&\qquad + kT\cdot \Pr\{T_{k-1}+T \le X < T_k\}\Bigg] \\
&= \sum_{k=1}^{\infty}\left[\int_{T_{k-1}}^{T_{k-1}+T}[t-(k-1)C]\,dF(t) + kT\int_{T_{k-1}+T}^{T_k} dF(t)\right] \\
&= \sum_{k=1}^{\infty}\left[\int_{T_{k-1}}^{T_{k-1}+T}\overline{F}(t)\,dt + (k-1)T\overline{F}(T_{k-1}) - kT\overline{F}(T_k)\right] \\
&= \sum_{k=1}^{\infty}\left[\int_{T_{k-1}}^{T_{k-1}+T}\overline{F}(t)\,dt\right]. \qquad (2.1)
\end{aligned}
$$

Let RU_k and RS_k denote the mean times required by the recovery actions when the failure occurs in the kth normal operation and in the kth checkpoint generation, respectively. By means of recovery function $R(x)$, we have

$$RU_k = \int_{T_{k-1}}^{T_{k-1}+T} R(t-T_{k-1})\,dF(t); \quad k \ge 1, \qquad (2.2)$$

$$RS_k = \int_{T_{k-1}+T}^{T_k} R(T)\,dF(t); \quad k \ge 1. \qquad (2.3)$$

From the above equations, we can derive the expected recovery time $RC(T)$ as follows:

$$RC(T) = \sum_{k=1}^{\infty}[RU_k + RS_k]. \qquad (2.4)$$

We further have the availability for one cycle $A(T)$:

$$A(T) = \frac{U_T}{E[X] + RC(T)}, \tag{2.5}$$

since the expected time for one cycle consists of the mean time to the failure and the expected recovery time.

Let us next consider the recovery function $R(x)$. The recovery action after the system failure consists of UNDO operation and REDO operation. For UNDO, which rolls back the effects of all incomplete transactions, the log file must be scanned to the BOT record of the oldest incomplete transaction. The amount of log information to process UNDO operation depends on the number of pages which have been propagated to the secondary storage before the failure even though the pages have been modified by incomplete transactions. We can assume that the overhead for UNDO operation is constant, i.e., the time interval between the latest checkpoint and the system failure is independent of the degree of the above redundant information, since transactions are recognized as a complete transaction one after another. REDO operation is processed by applying the results of all complete transactions, collected in the log file, to the secondary storage of the database. The amount of data to be processed for REDO operation depends on the time interval between the latest checkpoint and the system failure. Then we assume the following definition for $R(x)$:

$$R(x) = \mu r x + b, \tag{2.6}$$

which is similarly to Gelenbe [4] and Sumita et al. [5], where b represents the overhead for UNDO operation, μ the relative share of update transactions, and r the ratio of the overhead for a transaction in REDO operation to the overhead for a transaction in normal operation.

Substituting Equation (2.6) into Equation (2.4), we obtain the expected recovery time, $RC(T)$:

$$RC(T) = \mu r \sum_{k=1}^{\infty} [\int_{T_{k-1}}^{T_{k-1}+T} \overline{F}(t)\, dt - T\overline{F}(T_k)] + b. \tag{2.7}$$

Equations (2.5) and (2.7) are our new results for evaluating a recovery action in case where the failure rate of the system changes with time.

From these results, we can evaluate $RC(T)$ and $A(T)$ numerically, assuming an appropriate distribution to $F(t)$. We perform such evaluation in Subsect. 2.1.5. However, we first examine the behavior under the assumption that the time to the system failure obeys an exponential distribution with mean $1/\lambda$, i.e.,

$$F(t) = 1 - e^{-\lambda t}; \quad \lambda > 0. \tag{2.8}$$

We can obtain simpler forms in Equations (2.5) and (2.7). Chandy et al. [3], Gelenbe [4] and Young [6] also assumed Poisson failures, that is, the time to failure obeys an exponential distribution, which has the memoryless property. Hence we have

$$RC(T) = \frac{\mu r \left[\frac{1}{\lambda}(1 - e^{-\lambda T}) - Te^{-\lambda(T+C)}\right]}{1 - e^{-\lambda(T+C)}} + b, \tag{2.9}$$

and

$$A(T) = \frac{\frac{1}{\lambda}(1-e^{-\lambda T})}{\mu r\left[\frac{1}{\lambda}(1-e^{-\lambda T}) - Te^{-\lambda(T+C)}\right] + (\frac{1}{\lambda}+b)(1-e^{-\lambda(T+C)})}. \qquad (2.10)$$

We can explicitly obtain Equation (2.10) which enables us to derive analytically the optimum checkpoint interval T^* with the maximum availability $A(T^*)$. That is, we can derive the non-linear equation $\frac{d}{dT}A(T) = 0$ for T, which is restated as

$$\frac{1}{\mu r}(1+b\lambda)(e^{\lambda C}-1) + 1 - \lambda T - e^{-\lambda T} = 0. \qquad (2.11)$$

Consider applying the second order series approximation

$$e^{-\lambda T} \simeq 1 - \lambda T + \frac{\lambda^2}{2}T^2, \qquad (2.12)$$

to Equation (2.11). A truncation error remains less than $\mid (-\lambda T)^3/3! \mid$ in Equation (2.12). In the ordinary case, we can expect the approximation to be good one, since $T \ll E[X] = 1/\lambda$, i.e., $\lambda T \ll 1$ as we show in the next section. We can obtain T^* as follows:

$$T^* = \frac{1}{\lambda}\sqrt{\frac{2(1+b\lambda)(e^{\lambda C}-1)}{\mu r}}. \qquad (2.13)$$

This formula of the evaluation model for a database recovery action is a generalized version of the model for a endless job processing presented by Young [6], since it coincides with that obtained by Young when $\mu r = 1$, $b = 0$, and we apply the first order approximation to $e^{\lambda C}$.

2.1.5 Numerical Illustrations

In this subsection, we compute numerically the availability for one cycle, the optimum checkpoint intervals and the overhead for the recovery action, obtained in the preceding section. In the following, let us take up the robust numerical results which hold even in the case where the parameters vary in the ordinary range.

In many applications, we are interested in the early failure period of system operation, in which the time to the failure obeys a DFR distribution, and the random failure period, in which the time to the failure obeys a constant failure rate (CFR) distribution. Thus, we assume that the arbitrary failure-time distribution is a Weibull distribution:

$$F(t) = 1 - e^{-(\eta t)^m}; \quad \eta, m > 0, \tag{2.14}$$

where m and η are called the shape and scale parameters, respectively. The Weibull distribution gives a reasonable description of the above periods, since it is DFR for $0 < m < 1.0$ and CFR for $m = 1.0$. Note that the CFR distribution for $m = 1.0$ is an exponential distribution.

The system failures are principally caused by a software error of the DBMS or an operating system, a hardware failure, an unaccustomed operation by the system operator, and so on. The frequency of the failure is affected by the stability of the DBMS and the operational environment, and is assumed to be several times a week, like in the Reference [2]. Evaluating the early phase of the system operation, we assume that the occurrence of the failure is a little more frequent than is generally thought.

Figure 2.2 shows the dependence of checkpoint interval T on $A(T)$ for $m = 0.5$ and 1.0, where $E[X] = 10^4, 2.5 \times 10^4, 5 \times 10^4$[sec], $C = 2$[sec], $\mu = 0.8, r = 1.5$, and $b = 2$[sec]. It is obvious that the critical value of T, i.e., the optimum checkpoint interval, is yielded by means of the trade-off between the overhead for checkpoint generations and the overhead for recovery actions. Table 2.1 shows the optimum checkpoint interval T^* and $A(T^*)$ for each parameter in Fig. 2.2. We can see from these results that there is no great difference between the value of T^* or $A(T^*)$ for $m = 0.5$ and that for $m = 1.0$, and when T increases, the decrease in $A(T)$ for $m = 0.5$ is less than that of $m = 1.0$, in any $E[X]$. On the whole, the impact of $E[X]$ on the evaluation is greater than that of a shape of the failure time distribution. The more $E[X]$ increases, the more T^* and $A(T^*)$ increase. This fact shows clearly that if the system failure is a rare event, checkpoint generations should also be rare operations.

In the case of $m = 1.0$, we can use the analytical result of the formula (2.13) and obtain numerical results as $T^* = 183, 289, 408$ for $E[X] = 10^4, 2.5 \times 10^4, 5 \times 10^4$, respectively. These results show that our approximate formula is sufficiently precise.

Let us next discuss the optimum checkpoint interval T^* and $A(T^*)$ in varying other parameters. Figure 2.3 shows the dependence of T on $A(T)$,

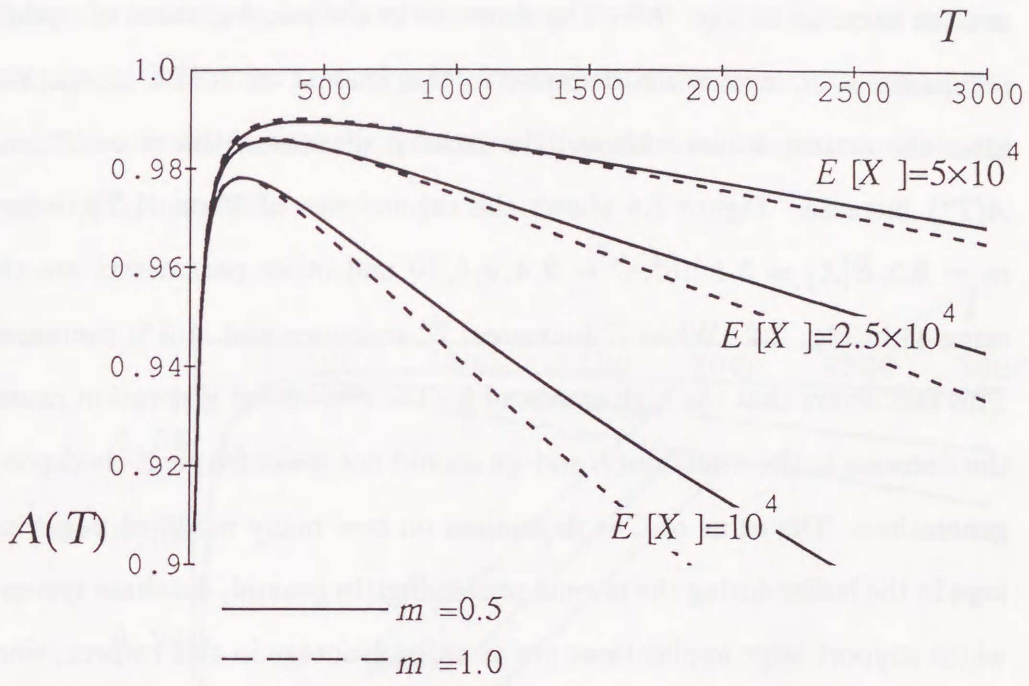

Figure 2.2: The dependence of T on $A(T)$ for $m = 0.5$ and 1.0 in varying $E[X]$ ($\mu = 0.8, r = 1.5, b = 2, C = 2$).

Table 2.1: T^* and $A(T^*)$ for each parameter in fig. 2.2.

m	$E[X]=10^4$		$E[X]=2.5\times10^4$		$E[X]=5\times10^4$	
	T^*	$A(T^*)$	T^*	$A(T^*)$	T^*	$A(T^*)$
1.0	184	0.978	288	0.986	408	0.990
0.5	194	0.978	305	0.985	426	0.989

where $m = 0.5, E[X] = 5 \times 10^4, \mu = 0.2, 0.5, 0.8$ and other parameters are the same as in Fig. 2.2. The decrease in the relative share of update transactions, μ, causes the decrease in the amount of REDO operations after the system failure. Hence, the more μ decreases, the more T^* and $A(T^*)$ increase. Figure 2.4 shows the dependence of T on $A(T)$, where $m = 0.5, E[X] = 5 \times 10^4, C = 2, 4, 6, 8, 10$ and other parameters are the same as in Fig. 2.2. When C increases, T^* increases and $A(T^*)$ decreases. This fact shows that the high overhead for the checkpoint generation causes the decrease in the availability, and we should not make frequent checkpoint generations. The value of C is dependent on how many modified pages are kept in the buffer during the normal processing. In general, database systems which support large applications are at a disadvantage in this respect, since its large buffer tends to have a great number of modified pages. We further obtain numerical results in varying b, the overhead for UNDO operation, e.g., b=0, 2, 4, 6, 8, 10. (Note that in the case of b=0, UNDO operation is not required.) However, the difference between these results is fairly small. The overhead for UNDO operation has almost no effect upon the evaluation of the recovery action with periodical checkpoint generations, since the total quantity of UNDO operation is far less than that of REDO operation.

Determining a checkpoint interval T, we can evaluate the expected recovery time $RC(T)$, and can limit the time for the recovery actions to $R(T)$ at the worst. Figure 2.5 show the dependence of T on $RC(T)$ and $R(T)$, where

$m = 0.5, E[X] = 5 \times 10^4$, and other parameters are the same as Fig. 2.2. From a practical point of view, the recovery actions may be bounded in time. We should determine a checkpoint interval with the maximum availability under the condition that the recovery actions must be completed within the appointed time limit.

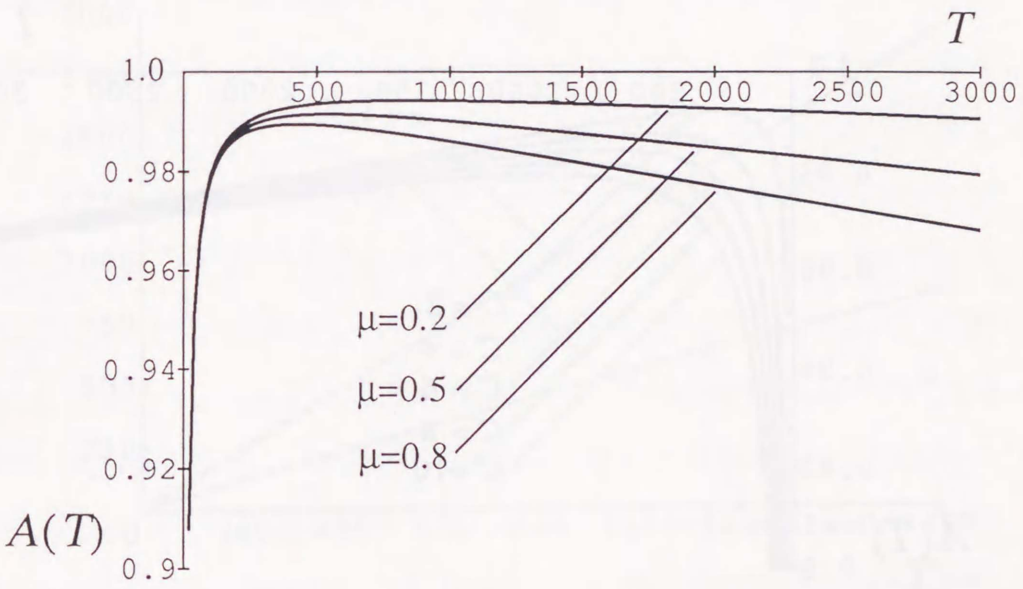

Figure 2.3: The dependence of T on $A(T)$ in varying μ $(m = 0.5, E[X] = 5 \times 10^4, r = 1.5, b = 2, C = 2)$.

2.1.6 Concluding Remarks

In this section, we have discussed the evaluation for a database recovery action with the periodical checkpoint generations. We have proposed a new model to evaluate the recovery action, where it is defined that one cycle

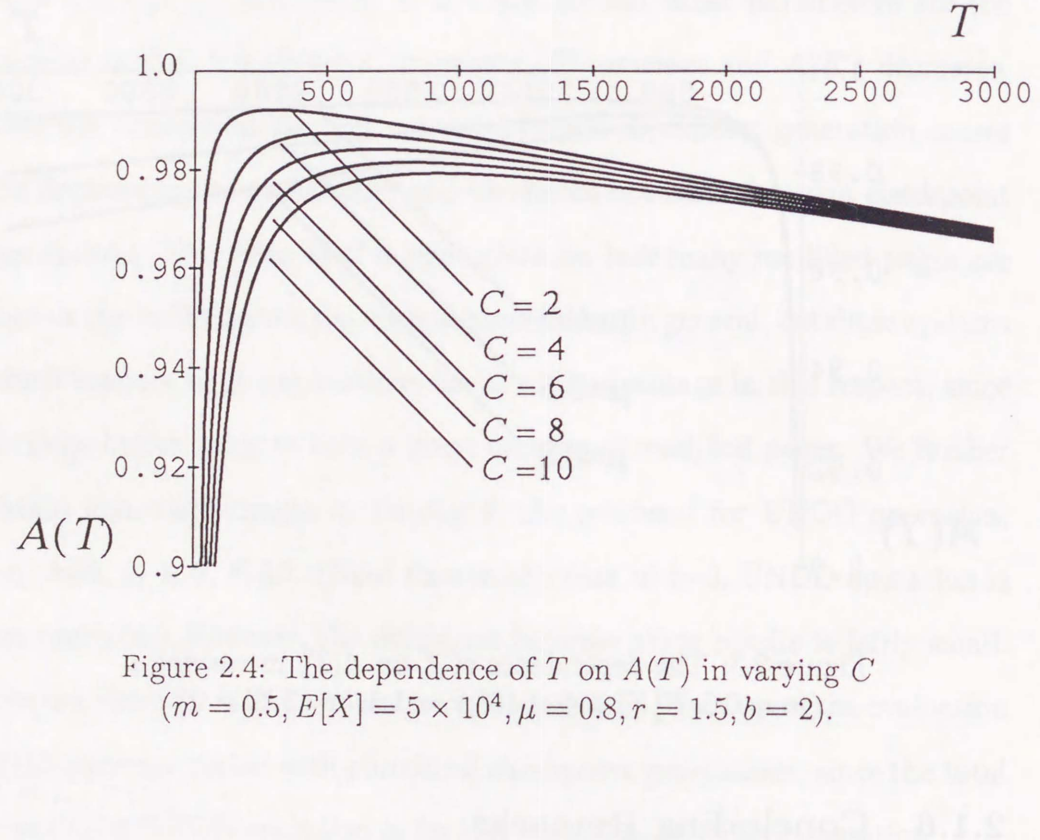

Figure 2.4: The dependence of T on $A(T)$ in varying C $(m = 0.5, E[X] = 5 \times 10^4, \mu = 0.8, r = 1.5, b = 2)$.

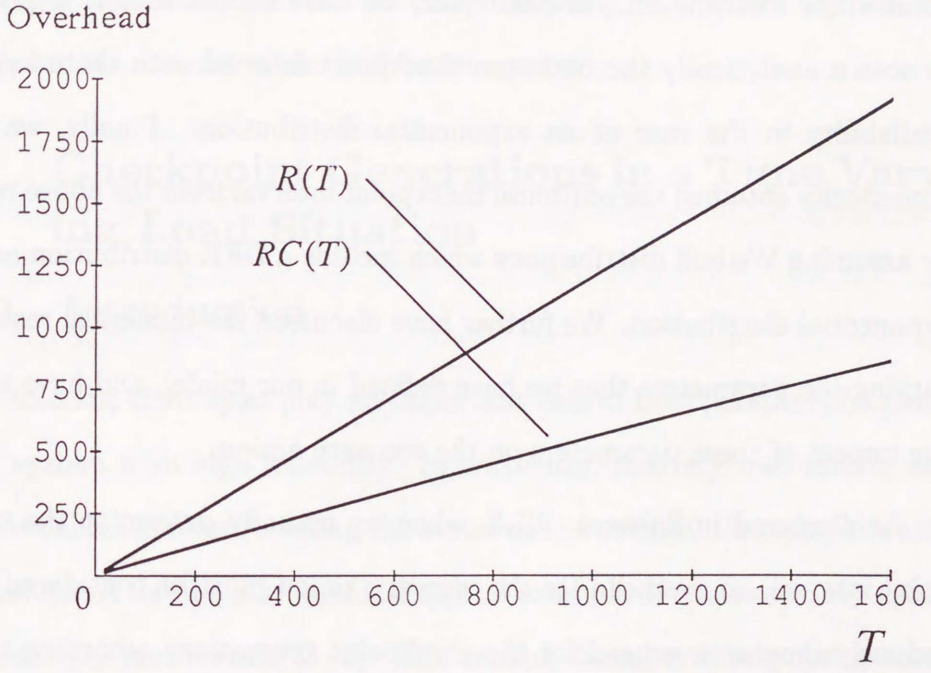

Figure 2.5: The dependence of T on $RC(T)$ and $R(T)$ $(m = 0.5, E[X] = 5 \times 10^4, \mu = 0.8, r = 1.5, b = 2, C = 2)$.

is the interval from the start of the system operation to the completion of recovery action after the failure in order to treat the changing of a failure rate of the system with time. We have obtained the expected recovery time and the availability for one cycle, under the assumption of an arbitrary failure-time distribution. In particular, we have shown that it is possible to obtain analytically the optimum checkpoint interval with the maximum availability in the case of an exponential distribution. Finally, we have numerically obtained the optimum checkpoint interval from the above results by assuming Weibull distributions which include a DFR distribution and an exponential distribution. We further have discussed the numerical results in varying the parameters that we have defined in our model, and have shown the impact of these parameters on the recovery action.

As discussed in Subsect. 2.1.5, when we actually determine the checkpoint interval, an overhead for the recovery action must be considered more seriously than an overhead for the checkpoint generations according to circumstances. For instance, an on-line banking system should be recovered after the system failure within a few minutes. If the recovery action takes more than an appointed time limit, we should make the checkpoint interval short so as to restrict the recovery time to the appointed time limit, even though the checkpoint interval maximizes the availability of our model.

Demand of operational environment which support large applications, such as OLTP, has an increasing tendency every year. The recovery techniques applied to the DBMS are more and more important. In these large systems, having large buffer capacity, the performance during the normal operation is influenced considerably by the overhead for the checkpoint generations. Therefore, database buffer management is important to system operation. In future, we have to study modeling of the recovery actions taking further account of the buffer management.

2.2 Checkpoint Generations in a Time Varying Load Situation

2.2.1 Introduction

Fault tolerant techniques play an important role in the operation of a computer system with high reliability. In particular, recovery mechanisms are indispensable for reconstructing the states of the computation after the system failure. A database system is a typical example of what seriously needs such recovery mechanisms [1, 2]. This section discusses checkpoint generations for a recovery mechanism on large applications of database systems.

When a system failure makes update information in the database buffer lost, the recovery action consists of two operations. One is *UNDO operation* which rolls back the effects of all incomplete transactions from the database, and the other is *REDO operation* which reflects the results of all complete transactions in the database (see [2]). In general, we execute REDO operation from the latest checkpoint instead of the starting point of the system operation. Generating a checkpoint implies that the update information in the buffer is collected in a stable secondary storage. It is important to decide the effective checkpoint generations. If we generate checkpoints frequently, we must incur large overhead for checkpoint generations, and conversely, if we generate few checkpoints, we must incur large overhead for recovery actions after the system failures. We should, therefore, decide checkpoint generations considering the trade-off between the two overheads above.

Several studies of deciding checkpoint generations have been discussed, which are the components of general recovery mechanisms including a database recovery. Young [6] derived the optimum checkpoint interval for the computation restart after the system failure. Chandy et al. [3] and Gelenbe [4] discussed evaluation models for database recovery and the generalized forms of the optimum checkpoint interval maximizing the system availability or the overhead during the normal operation. In these previous works, the failure rate of the system is assumed to be constant. We have proposed a model for evaluating the database recovery action in case where the failure rate of the system changes with time in the previous section. While these efforts yield the optimum checkpoint interval measured in unit of time, some models deal with the checkpoint interval measured in other quantity to describe the recovery mechanisms more reasonably. Reuter [7] considered the models to evaluate the transaction throughput as a performance measure for the database recovery mechanisms of the taxonomy in the reference [2], where the checkpoint interval is measured in unit of block transfers. Toueg and Babaoğlu [9] derived an algorithm which minimizes expected execution times of tasks placing checkpoints between two consecutive tasks with very general assumptions. Koren et al. [10] also discussed the model which minimizes the average time per instruction as a function of the number of instruction retries and the checkpoint interval measured in the number of the instructions, assuming the constant failure rate.

In this section, we propose a new model to determine the checkpoint generations for the database recovery. We consider that the transaction arrival rate and the failure rate of the system vary with time. The algorithm above derived by Toueg and Babaoğlu [9] seems to give a reasonable description of such situations. However, the dynamic programming algorithm, which yields the optimum sequence of checkpoints, is not suitable for large applications since the number of the transactions is expected to reach a great deal between the successive checkpoint generations. One of the primary interests in our model is that the transaction arrival rate, i.e., the load of the system changes with time in a shape of a cycle (e.g., a day) as an illustration of Fig. 2.6. In this case, we can see that the constant checkpoint interval measured in unit of time is not pertinent, since the failure rate of the system and the overhead for the recovery action obviously seems to vary with the load of the system. Taking account of these situations the third model exhibited by Chandy in the Reference [3] yields the problem of finding the shortest route of the graph whose nodes correspond to the beginning of intervals divided into from a cycle.

We derive an analytically efficient result by means of a simpler model. Occurrence of the failure and checkpoint generations are estimated by unit of update pages in the database buffer instead of time. We further regard the cumulative update of pages as a continuous quantity. Assume that the failure rate of the system (as a function of the cumulative update of pages) is dependent on the transaction arrival rate at which the corresponding page is updated, and the failure mode of a cycle is described as consisting of phases, e.g., as shown in Fig. 2.7. The optimum checkpoint generations are derived as the sequence measured in the cumulative update minimizing the expected total overhead to completion of a phase, where the checkpoint interval changes with the failure rate of the system.

In the following subsection, we define our new model introducing a density of checkpoint generations and several assumptions. Subsection 2.2.3 discusses the analysis of the model. The expected total overhead to completion of a phase is derived. We obtain the density of checkpoint generations minimizing the total overhead, which yields the optimum sequence of checkpoint generations. Moreover, the above total overhead and density are replaced by new forms assuming concrete overhead functions. We next show the results in case where the cumulative update to the system failure obeys a Weibull distribution. Subsection 2.2.4 gives numerical examples for

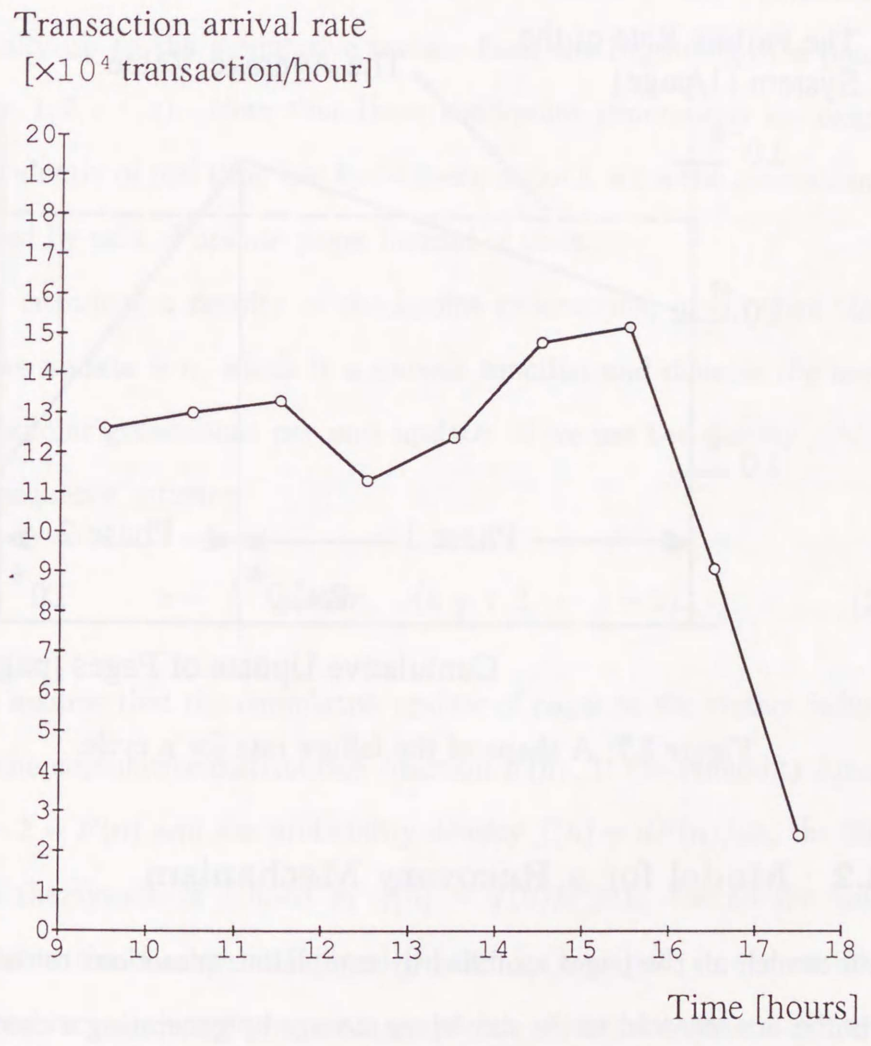

Figure 2.6: A sample function of transaction arrival rate for a cycle.

our analyses under the assumption that the failure rate is described as the shape of phases in Fig. 2.7.

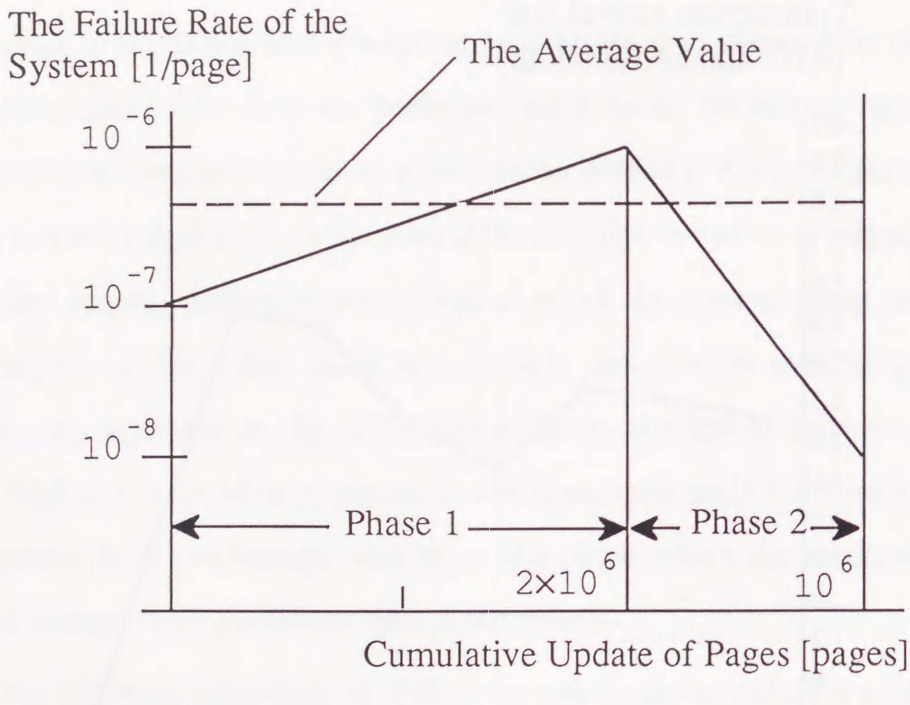

Figure 2.7: A shape of the failure rate for a cycle.

2.2.2 Model for a Recovery Mechanism

In our model, all the pages modified by complete transactions remaining in the buffer are reflected to the secondary storage by generating a checkpoint. While the cumulative update of pages in the buffer is a discrete quantity obviously, we can regard it as continuous, since we consider a great number of update pages such as thousands or tens of thousands of pages. A phase we deal with completes when the cumulative update reaches to N, with the sth checkpoint generation. Let $\{n_1, n_2, \cdots, n_{s-1}, n_s(= N)\}$ be the sequence of checkpoint generations, where each checkpoint is generated sequentially up to the cumulative update from the beginning of a phase to $n_k (k = 1, 2, \cdots, s)$. Note that these checkpoint generations are executed independently of real time lost by recovery actions, since the generations are managed by unit of update pages instead of time.

We introduce a density of checkpoint generations, $g(n)$, when the cumulative update is n, which is a smooth function and denotes the number of checkpoint generations per unit update. If we use the density $g(n)$, the above sequence satisfies:

$$k = \int_0^{n_k} g(n)dn, \quad (k = 1, 2, \cdots, s-1). \tag{2.15}$$

We assume that the cumulative update of pages to the system failure X obeys the cumulative distribution function $F(n)$. If the reliability function $\bar{F}(n) = 1 - F(n)$ and the probability density $f(n) = dF(n)/dn$, the failure rate of the system is defined by $\gamma(n) = f(n)/\bar{F}(n)$. For all the failures occurred in the checkpoint interval $(n_{k-1}, n_k], (k = 1, 2, \cdots, s; n_0 = 0)$, we make recovery actions from the state of kth checkpoint generation to the consistent states which had been constructed just before those failures. We consider that checkpoint generations and recovery actions never cause the system failure and never change the failure rate of the system.

The expected total overhead to completion of a phase, $L(N, g(n))$, consists of the overhead for checkpoint generations to completion of a phase and the expected overhead for recovery actions to completion of a phase. In order to derive these overheads, we introduce the overhead for the kth checkpoint generation, $H_c(n_k - n_{k-1})$, and the overhead for a recovery action, $H_r(n - n_l)$, in case $X = n$ and the latest checkpoint generation is the lth one.

2.2.3 Analysis

General Analysis

Let us derive the optimum sequence $\{n_1^*, n_2^*, \cdots, n_{s-1}^*, n_s\}$ which minimizes the expected total overhead to completion of a phase from the assumptions above.

First, the overhead for checkpoint generations to completion of a phase is obtained as follows by using the density of checkpoint generations:

$$\sum_{k=1}^{s} H_c(n_k - n_{k-1}) = \int_0^N H_c(g(n)^{-1})g(n)dn. \tag{2.16}$$

We next derive the expected overhead for recovery actions to completion of a phase. If $X = n$, the overhead for recovery actions between two successive checkpoint generations is approximately given by

$$\begin{aligned} &\int_{n_{k-1}}^{n_k} H_r(n - n_{k-1})\gamma(n) \\ &\simeq \int_{n_{k-1}}^{n_k} H_r(\frac{n_k - n_{k-1}}{2})\gamma(n)dn \end{aligned}$$

$$= H_r(\frac{n_k - n_{k-1}}{2}) \int_{n_{k-1}}^{n_k} \gamma(n)dn, \quad (k = 1, 2, \cdots, s), \quad (2.17)$$

where we consider the overhead for a recovery action to be equal to the overhead after a system failure in the middle of the checkpoint interval, in average, similarly to Reuter [7]. This approximation can be expected to be a good one, since we are estimating the mean value of the total overhead. Thus, we can obtain the expected overhead for recovery actions to completion of a phase:

$$\sum_{k=1}^{s} \int_{n_{k-1}}^{n_k} H_r(n - n_{k-1})\gamma(n)dn$$

$$\simeq \sum_{k=1}^{s} H_r(\frac{n_k - n_{k-1}}{2}) \int_{n_{k-1}}^{n_k} \gamma(n)dn$$

$$= \int_0^N H_r(\frac{1}{2}g(n)^{-1})\gamma(n)dn. \quad (2.18)$$

From Equations (2.16) and (2.18), we have the expected total overhead to completion of a phase:

$$L(N, g(n)) = \int_0^N \Big[H_c(g(n)^{-1})g(n) + H_r(\frac{1}{2}g(n)^{-1})\gamma(n)\Big]dn. \quad (2.19)$$

We obtain the density of checkpoint generations, $g(n)$, minimizing the functional $L(N, g(n))$. This is a problem of calculus of variations in which $g(n)$ is the unknown function. Euler's equation implies

$$H_c(g(n)^{-1}) - g(n)^{-1}H_c'(g(n)^{-1}) - \frac{1}{2}g(n)^{-2}H_r'(\frac{1}{2}g(n)^{-1})\gamma(n) = 0. \quad (2.20)$$

Applying concrete overhead functions $H_c(\cdot)$ and $H_r(\cdot)$, and solving Equation (2.20) yield the density $g(n)$. Substituting $g(n)$ into Equation (2.15) enables us to derive the optimum sequence $\{n_1^*, n_2^*, \cdots, n_{s-1}^*, n_s\}$.

Overhead Functions

Let us introduce concrete overhead functions to obtain the density $g(n)$ based on the analytical results above. In large applications of database systems, we can assume the overhead function for a checkpoint generation to be the simplest form:

$$H_c(x) = h_c, \tag{2.21}$$

that is, the overhead for a checkpoint generation is always constant and independent of the checkpoint interval (see [2]). We further assume the overhead function for a recovery action:

$$H_r(x) = h_r x + h_u, \tag{2.22}$$

where h_u is the constant overhead for UNDO operation and h_r is the overhead for REDO operation per unit update of pages corresponding to the forms of [3] and [4]. From Equation (2.19), the expected total overhead to completion of a phase is given by

$$L(N, g(n)) = \int_0^N \Big[h_c g(n) + (\frac{h_r}{2g(n)} + h_u)\gamma(n)\Big] dn. \tag{2.23}$$

We further obtain Euler's equation from Equation (2.20):

$$h_c - \frac{h_r}{2} g(n)^{-2}\gamma(n) = 0. \tag{2.24}$$

Solving Equation (2.24) with respect to $g(n)$ yields:

$$g(n) = \sqrt{\frac{h_r}{2h_c}\gamma(n)}. \tag{2.25}$$

A Case of Weibull Distribution

We next discuss a case where the cumulative update of pages to the system failure obeys the Weibull distribution:

$$F(n) = 1 - e^{-(\eta n)^m}, \quad (\eta > 0, m > 0). \tag{2.26}$$

The weibull distribution is able to give a reasonable description of several failure modes, in which the failure rates change with the time variables, by varying the parameters. The parameters η and m are called the scale and shape parameters, respectively. We have $\bar{F}(n) = e^{-(\eta n)^m}$, $f(n) = m\eta^m n^{m-1} e^{-(\eta n)^m}$ and $\gamma(n) = m\eta^m n^{m-1}$.

From Equation (2.25), the density of checkpoint generations is given by

$$g(n) = \sqrt{\frac{h_r m\eta^m n^{m-1}}{2h_c}}. \tag{2.27}$$

Moreover, substituting $g(n)$ from Equation (2.27) into Equation (2.23) yields the expected total overhead to completion of a phase:

$$L(N, g(n)) = \frac{2\sqrt{2h_c h_r m\eta^m}}{m+1} N^{\frac{m+1}{2}} + h_u(\eta N)^m. \tag{2.28}$$

From Equations (2.15) and (2.27), we can explicitly obtain the optimum sequence as follows:

$$n_k^* = (m+1)^{\frac{2}{m+1}} \Big(\frac{h_c}{2h_r m\eta^m}\Big)^{\frac{1}{m+1}} k^{\frac{2}{m+1}}, \quad (k = 1, 2, \cdots, s-1). \tag{2.29}$$

We can see that the interval between checkpoint generations increases with the cumulative update for $0 < m < 1$ and decreases for $1 < m$. In particular, in case of $m = 1$, $F(n)$ is an exponential distribution. We have the constant intervals between checkpoint generations:

$$n_k^* - n_{k-1}^* = \sqrt{\frac{2h_c}{h_r \eta}}, \quad (k = 1, 2, \cdots, s-1), \tag{2.30}$$

which coincides with the formula obtained by Young [6] when we regard n as the time variable and $h_r = 1$.

2.2.4 Numerical Illustrations

Let us numerically compute the sequence of checkpoint generations by assuming the phases as shown in Fig. 2.7. If the failure rate $\gamma(n)$ is described as the function of the first degree, i.e., $\gamma(n) = vn + w$, the optimum sequence of checkpoint generations is given by

$$n_k^* = \frac{1}{v}\Big[\frac{3v}{2}\sqrt{\frac{2h_c}{h_r}} \cdot k + w^{\frac{3}{2}}\Big]^{\frac{2}{3}} - \frac{w}{v}, \quad (k = 1, 2, \cdots, s-1), \tag{2.31}$$

from Equations (2.15) and (2.25). We further have the expected total overhead from Equation (2.23) as follows:

$$L(N, g(n)) = \frac{2\sqrt{2h_c h_r}}{3v}\Big\{(vN + w)^{\frac{3}{2}} - w^{\frac{3}{2}}\Big\} + h_u(\frac{v}{2}N^2 + wN). \tag{2.32}$$

Let $\{n_1', n_2', \cdots, n_{s-1}', n_s\}$ be the sequence of checkpoint generations in case where the constant failure rate η_c, that is the average value of the failure rate of Fig. 2.7, is used instead of the failure rate of the phase

1 or the phase 2 to obtain the density $g(n)$. Table 2.2 shows the optimum sequence $\{n_1^*, n_2^*, \cdots, n_{s-1}^*, n_s\}$ for the phase 1, and the sequence $\{n'_1, n'_2, \cdots, n'_{s-1}, n_s\}$, where $v = (10^{-6} - 10^{-7})/(2 \times 10^6)$, $w = 10^{-7}, N = 2 \times 10^6, h_c = 5$[sec], $h_r = 0.1$[sec] and $h_u = 5$[sec]. Figure 2.8 illustrates the relation between the sequence and the density of checkpoint generations $g(n)$. Table 2.3 and Fig. 2.9 similarly show the results for the phase 2, where $v = (10^{-8} - 10^{-6})/10^6, w = 10^{-6}, N = 10^6$ and the other parameters are the same as in Table 2.2. Note that the checkpoint interval is decreasing with the cumulative update in case of the phase 1, since the failure rate is increasing. Conversely, the interval is increasing with the cumulative update in case of the phase 2, since the failure rate is decreasing.

We next discuss comparisons between the expected total overhead by the optimum sequence $\{n_1^*, n_2^*, \cdots, n_{s-1}^*, n_s\}$ and the one by the sequence $\{n'_1, n'_2, \cdots, n'_{s-1}, n_s\}$ assuming the failure rate is described as the phase 1 or the phase 2. Let L_p denote the expected total overhead by the optimum sequence which is obtained by Equation (2.32). Furthermore, let L_c denote the expected total overhead by the sequence $\{n'_1, n'_2, \cdots, n'_{s-1}, n_s\}$. We can obtain L_c from Equation (2.19) in which $g(n)$ is derived by the constant failure rate above although $\gamma(n)$ is the failure rate of the phase 1 or the phase 2. Table 2.4 shows the gain of L_p to L_c, $((L_c - L_p)/L_c) \times 100[\%]$, for the phase 1 and the phase 2, where all parameters are the same as in Tables 2.2 and 2.3, and the average value of the failure rate is calculated as

Table 2.2: The sequences of checkpoint generations for the phase 1.

k	$n_k{}^*$	$n_k{}'$
	$[\times 10^4$ pages]	
1	3.05	1.36
2	5.94	2.73
3	8.68	4.10
4	11.31	5.46
.	.	.
.	.	.
.	.	.
140	196.52	191.40
141	197.52	192.77
142	198.53	194.13
143	199.53	195.50
144	200.00	200.00

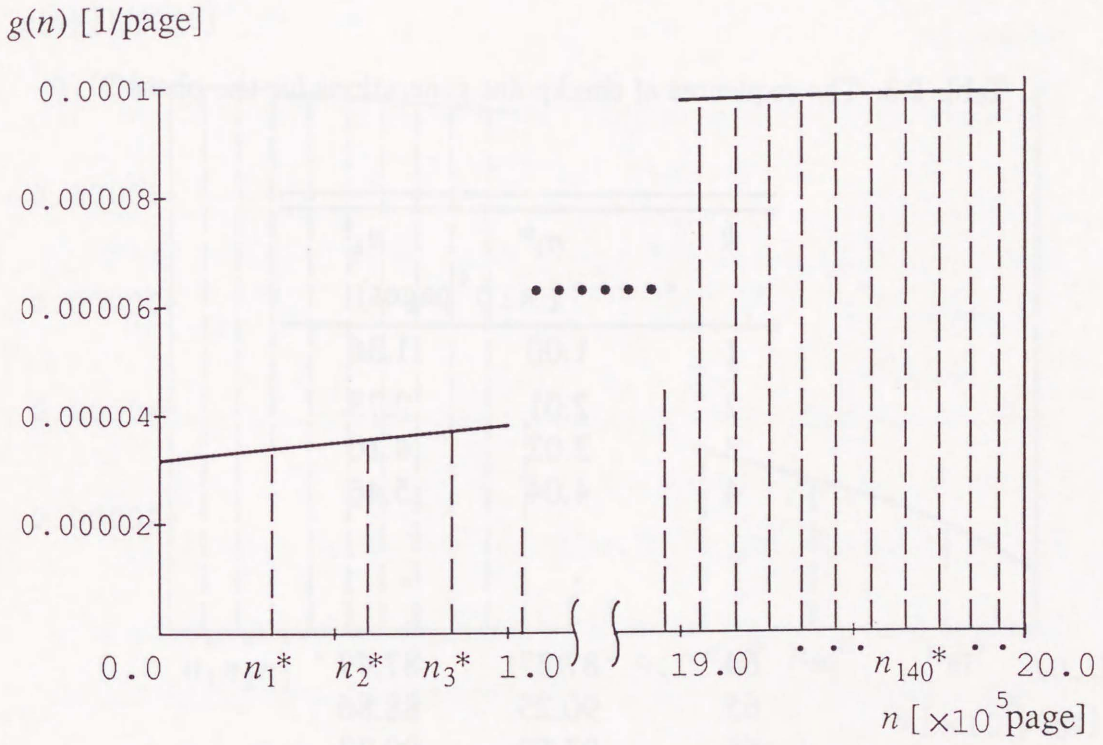

Figure 2.8: The illustration for the density and the sequence of checkpoint generations for the phase 1.

Table 2.3: The sequences of checkpoint generations for the phase 2.

k	n_k^*	n_k'
	[$\times 10^4$ pages]	
1	1.00	1.36
2	2.01	2.73
3	3.02	4.10
4	4.04	5.46
.	.	.
.	.	.
.	.	.
64	87.37	87.49
65	90.25	88.86
66	93.59	90.23
67	98.03	91.60
68	100.00	100.00

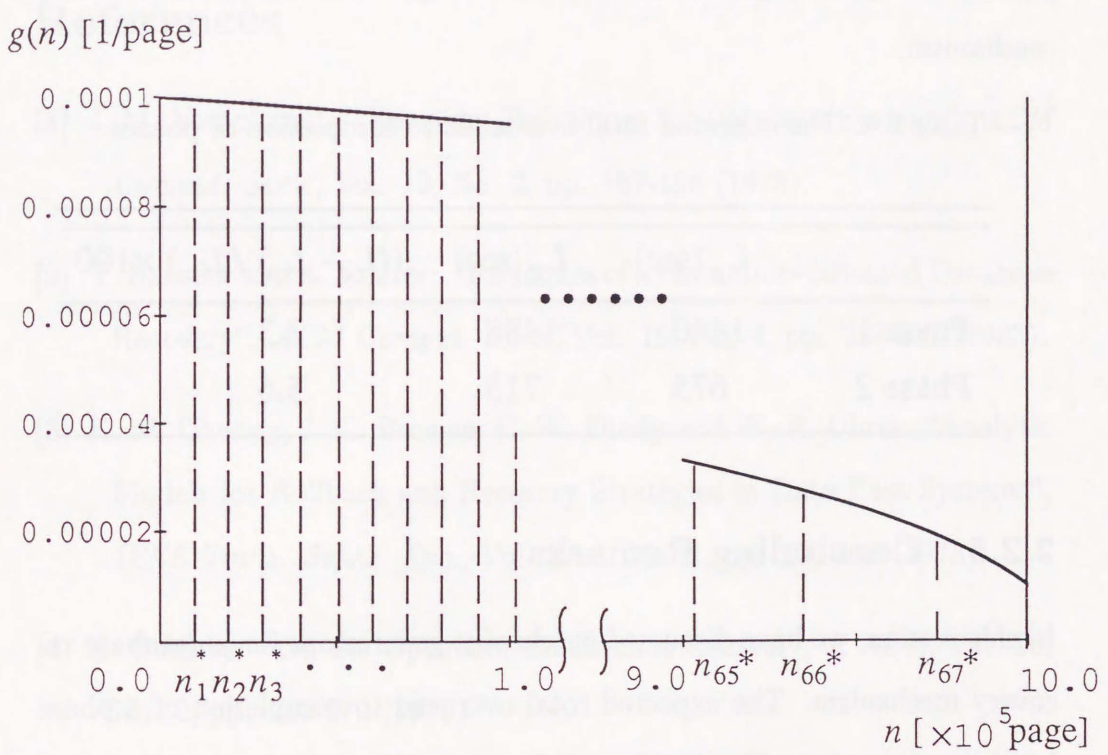

Figure 2.9: The illustration for the density and the sequence of checkpoint generations for the phase 2.

$\eta_c = 5.35 \times 10^{-7}$. It is evident that checkpoint generations by the optimum sequence is more effective than the other in either case. This fact implies that the sequence of checkpoint generations, varying its interval with the failure rate of the system, gives a reasonable strategy of the database recovery mechanism.

Table 2.4: The expected total overheads to completion of phases.

	L_p [sec]	L_c [sec]	$((L_c - L_p) / L_c) \times 100$
Phase 1	1440	1488	3.3
Phase 2	675	713	5.6

2.2.5 Concluding Remarks

In this section, we have discussed checkpoint generations for a database recovery mechanism. The expected total overhead to completion of a phase has been presented. The density of checkpoint generations has been analytically derived minimizing the total overhead, which yields the optimum sequence of checkpoint generations measured in unit of update pages. Finally, numerical examples for the results have been given in case where the failure rate of a phase is described as a linear shape.

The results presented in this paper are the analytical ones. Applying the appropriate failure rate and the parameters enable us to calculate the optimum sequence relatively easily. We can see that the sequence obtained is of great use for various kinds of failure modes and gives reasonable strategy for checkpoint generations as discussed by the numerical examples above.

References

[1] J. M. Verhofstadt: "Recovery Techniques for Database Systems", *ACM Comput. Surv.*, Vol. 10, No. 2, pp. 167-196 (1978).

[2] T. Haerder and A. Reuter: "Principles of Transaction-Oriented Database Recovery", *ACM Comput. Surv.*, Vol. 15, No. 4, pp. 287-317 (1983).

[3] K. M. Chandy, J. C. Browne, C. W. Dissly and W. R. Uhrig: "Analytic Models for Rollback and Recovery Strategies in Data Base Systems", *IEEE Trans. Softw. Eng.*, Vol. SE-1, No. 1, pp. 100-110 (1975).

[4] E. Gelenbe: "On the Optimum Checkpoint Interval", *J. ACM*, Vol. 26, No. 2, pp. 259-270 (1979).

[5] U. Sumita, N. Kaio and P. B. Goes: "Analysis of Effective Service Time with Age Dependent Interruptions and Its Application to Rollback Policy for Database Management", *Queueing Systems*, No. 4, pp. 193-212 (1989).

[6] J. W. Young: "A First Order Approximation to the Optimum Checkpoint Interval", *Comm. ACM*, Vol. 17, No.9, pp. 530-531 (1974).

[7] A. Reuter: "Performance Analysis of Recovery Techniques", *ACM Trans. Database Syst.*, Vol. 9, No. 4, pp. 526-559 (1984).

[8] S. M. Ross : *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco (1970).

[9] S. Toueg and Ö. Babaoğlu: "On the Optimum Checkpoint Selection Problem", *SIAM J. Comput.*, Vol. 13, No. 3, pp. 630-649 (1984).

[10] I. Koren, Z. Koren and S. Y. H. Su: "Analysis of a Class of Recovery Procedures", *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 703-712 (1986).

Chapter 3

Effects of Time Redundancy by Retries

3.1 Introduction

Fault-tolerance for a computer system is attained by some *redundancy*. We can classify redundancies into three types: hardware redundancy, software redundancy and time redundancy according to their actualization methods [1]. For instance, TMR (Triple Modular Redundancy) or graceful degradation techniques are good examples of hardware redundancy. It is similarly well-known that N-version programming and recovery blocks are based on the concept of software redundancy. Moreover, a typical example of time redundancy must be given by retries for instructions or I/O operations.

Time redundancy for a computer system is more generally available than hardware redundancy or software redundancy. In most cases, hardware redundancy and software redundancy require physical and/or logical resources allocated previously, e.g., processor units, alternate software modules or the subsystems. We can not always install these resources since they are dependent on system configuration and cause the increase of the hardware or software cost. On the other hand, few additional resources are needed for executing the time redundant methods.

In this chapter, we present a model for evaluating the improvement on system reliability by retries based on time redundancy. While retry procedures are incorporated into recovery mechanisms of almost all the systems for the reason above, there are few previous works of the evaluation modeling for retry procedures. It is interesting and important for a system design or maintenances to estimate the effects of time redundant techniques as well as hardware and software redundant techniques.

Retries are principally executed for the purpose of recovery from an *intermittent* failure [1, 2, 4]. The effect of this failure is temporary, and is caused by an error operation of electronic devices, a contact fault in a hardware and so on. On the other hand, a *permanent* failure is continuous and stable, and is caused by a damage of electronic devices or wiring in a hardware. In the actual computer systems, intermittent failures are said to be the most frequent failures. Thus, when a failure takes place, the repeated actions by retry procedures are greatly useful for the first step of recovery from the malfunctional state.

However, even though the retry succeeds and the system is restored to the correct state, information of the failure state is logged out to investigate the cause of the failure [2]. From the viewpoint of a maintenance, we should remove the intermittent failures masked by retries, some day, since the latent failures will bring on the same failures and may develop into the permanent failures. An intermittent failure is more difficult to detect and remove than a permanent failure, as intermittent failures are not reproducible. The maintenance is executed based on the log out information above.

The model we discuss here is concerned with the references [3-7]. A model for reliability analysis of a computer system with retry has been proposed in the reference [3], where the maintenance is carried out with the inspection interval of T, or with the number of successful retries K. Several models for intermittent failures have been discussed, which describe diagnosable systems [4] and maintenance strategies [5-7].

Taking account of the behavior of intermittent and permanent failures, we consider the evaluation model which manages the system maintenance with the prescribed number of successful retries. When a failure takes place, retry procedures are repeated m times at the maximum limit. If no retry succeeds within the limit, we identify the failure as a permanent failure and remove it by the maintenance. If any retry succeeds, we identify the failure as an intermittent failure and remove it by the maintenance after the successful retries are observed N times.

Our main interest in this model is to discuss the reliability evaluation in the transient-state as well as the steady-state. Evaluation in the steady-state is important for a long term operation of the systems. However, evaluation in the transient-state, such as an early phase of the system operation, is not always coincide with the one in the steady-state.

In the next section, we define the evaluation model introducing several assumptions. Section 3.3 analyzes the model by applying the Markov renewal processes [8, 9] to derive the availability and the mean time between failures in the steady-state. Moreover, a calculation method for the availability in the transient-state is described by continuous-time Markov chains and a randomization technique [10]. In Section 3.4, we discuss the numerical illustrations such as the numbers of successful retries with which the availabilities are maximized.

3.2 Model for a Retry Procedure

In our model, when a failure takes place, retries are executed m times at the maximum limit. If any retry succeeds within the limit, we identify the failure as an intermittent failure, and if no retry succeeds, we identify the failure as a permanent failure. We introduce a model described as follows for evaluating the effects of retry procedures.

(1) Consider a situation just after a start of system operation. We assume that the time to the first failure obeys an arbitrary distribution $A_0(t)$

with a finite mean a_0, and an overhead time for a retry, $B_0(t)$ with a finite mean b_0. Each retry succeeds with the probability r_{0j} ($j = 1, 2, \cdots, m$), and does not succeed with probability $\overline{r_{0j}}(= 1 - r_{0j})$. Taking account that the probability of success deteriorates according as the number of retries are increasing, we assume that $r_{01} > r_{02} > \cdots > r_{0m}$. If retries do not succeed m times, the failure is identified as a permanent failure and is removed by a maintenance. This maintenance incurs an overhead which obeys the arbitrary distribution $R_0(t)$ with a finite mean σ_0, and restore the failure mode of the system to the one at the start of system operation.

(2) Consider situations just after the successful retries are observed k times. We assume that the time to the failure obeys an arbitrary distribution $A_k(t)$ ($k = 1, 2, \cdots, N - 1$) with a finite mean a_k, and an overhead time for a retry, $B_k(t)$ with a finite mean b_k. Each retry succeeds with the probability r_{kj} ($r = 1, 2, \cdots, m$), and does not succeed with probability $\overline{r_{kj}}(= 1 - r_{kj})$. Taking account that the probability of success deteriorate according as the number of retries are increasing, we assume that $r_{k1} > r_{k2} > \cdots > r_{km}$. If retries do not succeed m times, the failure is identified as a permanent failure and is removed by a maintenance. This maintenance incurs an overhead time which obeys the arbitrary distribution $R_k(t)$ with a finite mean σ_k, and restores the

failure mode of the system to the one just after the successful retries are observed k times. We further describe the influence of failure latencies produced by successful retries as $a_0 > a_1 > \cdots > a_{N-1}$ and $r_{01} > r_{11} > \cdots > r_{N-1\ 1}$.

(3) If the successful retries are observed N times ($N \geq 1$), intermittent failures latent in the system are removed by a maintenance based on log-out information. This maintenance incurs an overhead time which obeys the distribution $Z_N(t)$ with a finite mean ϕ_N, and restores the failure mode of the system to the one at the start of system operation.

We define the following states which characterize the behavior of the system.

State 0_0 : State after the start of system operation.

State 0_j : State that the jth retry is executing after the failure on State 0_0 ($j = 1, 2, \cdots, m$).

State k_0 : State after the successful retries are observed k times ($k = 1, 2, \cdots, N-1$).

State k_j : State that the jth retry is executing after the failure on State k_0 ($k = 1, 2, \cdots, N-1; j = 1, 2, \cdots, m$).

State D_k : State that all m times retries do not success after the failure on State k_0, and the maintenance for the permanent failure is being executed $(k = 0, 1, 2, \cdots, N-1)$.

State D : State that successful retries are observed N times and the maintenance for the intermittent failures is being executed.

We redefine State k $(k = 0, 1, \cdots, N-1)$ which is composed of $m+1$ states, State k_0, State k_1, State k_2, ... and State k_m. We can formulate the model by a Markov renewal process with the states above. The state transition diagram is shown in Fig. 3.1.

3.3 Analysis

We first derive the availability, A_v, and the mean time between failures, MTBF, in the steady-state, applying Markov renewal processes theory [8, 9].

Let $F_k(t)$ denote the probability distribution function with which any retry succeeds within m times after the failure on State k $(k = 0, 1, \cdots, N-1)$, and $G_k(t)$ the probability distribution function with which no retry succeeds. We have the following equations in terms of convolutions:

$$F_k(t) = A_k(t) * \sum_{j=1}^{m}[\{C_k(j-1) - C_k(j)\}B_k^{(j)}(t)], \tag{3.1}$$

$$G_k(t) = A_k(t) * C_k(m) * B_k^{(m)}(t), \tag{3.2}$$

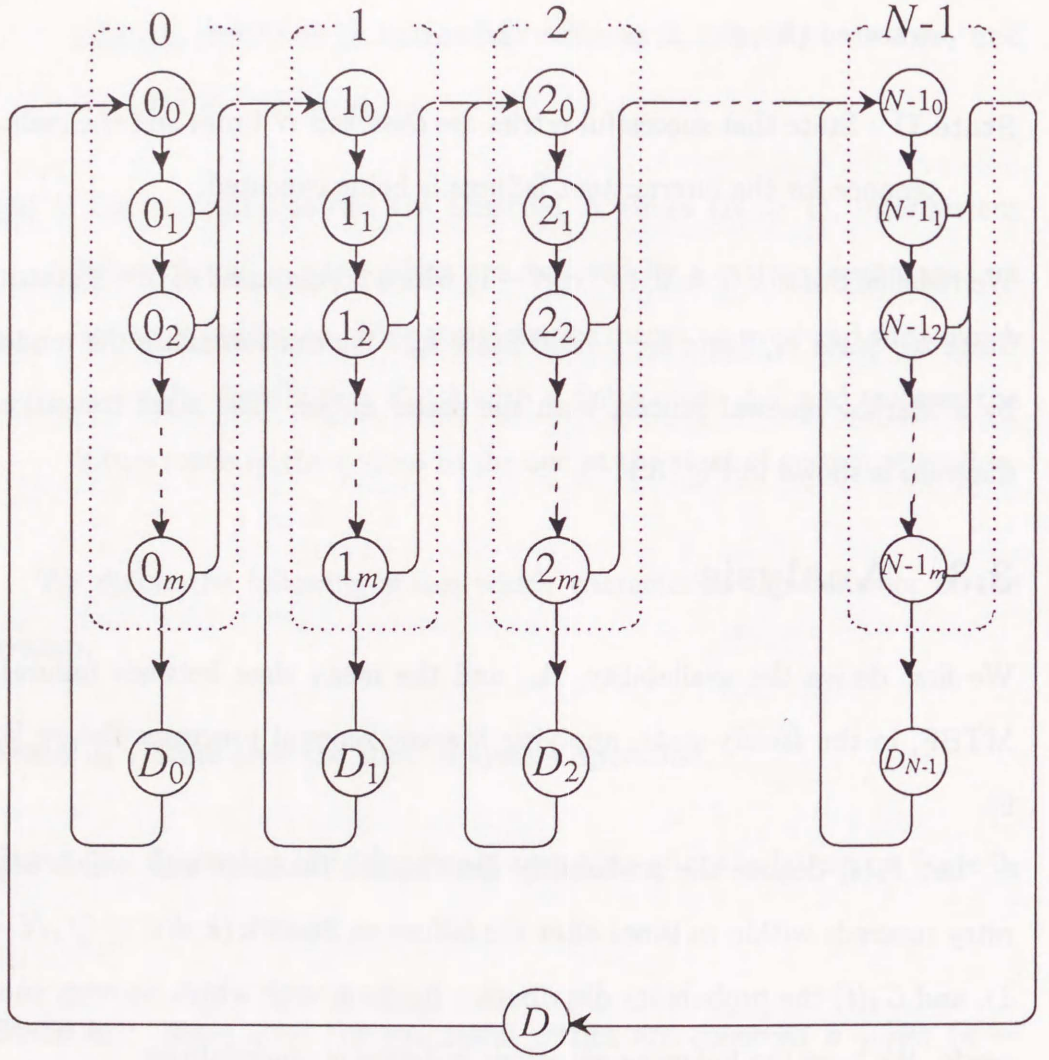

Figure 3.1: The state transition diagram of the computer system.

where

$$C_k(0) \equiv 1, \tag{3.3}$$

$$C_k(j) \equiv \bar{r}_{k1}\bar{r}_{k2}\cdots\bar{r}_{kj}, \quad (j = 0, 1, \cdots, m), \tag{3.4}$$

and $B_k^{(j)}(t)$ implies the j-fold convolution of $B_k(t)$ with itself.

Let $Q_{ij}(t)$ denote the transition probabilities from State i to State j for the Markov renewal process. We can define the following probabilities:

$$Q_{kk+1}(t) = F_k(t) \quad (k = 0, 1, \cdots, N-1), \tag{3.5}$$

$$Q_{kD_k}(t) = G_k(t) \quad (k = 0, 1, \cdots, N-1), \tag{3.6}$$

$$Q_{D_k k}(t) = R_k(t) \quad (k = 0, 1, \cdots, N-1), \tag{3.7}$$

$$Q_{N-1D}(t) = F_{N-1}(t), \tag{3.8}$$

$$Q_{D0}(t) = Z_N(t). \tag{3.9}$$

We also have $q_{ij}(s)$, the LS (Laplace-Stieltjes) transforms of these transition probabilities:

$$q_{kk+1}(s) = a_k(s)\sum_{j=1}^{m}[\{C_k(j-1) - C_k(j)\}b_k^j(s)] \quad (k = 0, 1, \cdots, N-1), \tag{3.10}$$

$$q_{kD_k}(s) = a_k(s)b_k^m(s)C_k(m) \quad (k = 0, 1, \cdots, N-1), \tag{3.11}$$

$$q_{D_k k}(s) = r_k(s) \quad (k = 0, 1, \cdots, N-1), \tag{3.12}$$

$$q_{N-1D}(s) = a_{N-1}(s)\sum_{j=1}^{m}[\{C_{N-1}(j-1) - C_{N-1}(j)\}b_{N-1}^j], \tag{3.13}$$

$$q_{D0}(s) = z_N(s), \tag{3.14}$$

where $a_k(s), b_k(s), r_k(s)$ and $z_N(s)$ are the LS transforms of $A_k(t), B_k(t), R_k(t)$ and $Z_N(t)$, respectively.

Let us derive the mean recurrence time l_{00} for State 0. If we define

$$u_j(t) = F_{j-1}(t) * \sum_{l=0}^{\infty}(G_j(t) * R_j(t))^{(l)} \quad (j = 1, 2, \cdots, N-1), \tag{3.15}$$

the recurrence time distribution for State 0 is obtained by

$$H_{00}(t) = G_0(t) * R_0(t) + \{u_1(t) * u_2(t) * \cdots * u_{N-1}(t)\} * F_{N-1}(t) * Z_N(t), \tag{3.16}$$

and its LS transform $h_{00}(s)$ yields l_{00} as follows:

$$\begin{aligned} l_{00} &= -\frac{d}{ds}h_{00}(s)|_{s=0} \\ &= \{1 - C_0(m)\}\{\sum_{k=0}^{N-1}\frac{a_k + b_k\sum_{j=1}^{m}C_k(j)}{1 - C_k(m)} \\ &\qquad + \sum_{k=0}^{N-1}\frac{C_k(m)\sigma_k}{1 - C_k(m)} + \phi_N\}. \end{aligned} \tag{3.17}$$

From the above results, we can derive M_j, the expected number of visits to State j $(j = 0, 1, \cdots, N-1, D_0, D_1, \cdots, D_{N-1}, D)$ per unit time in the steady-state. Let $M_j(t)$ denote the expected number of visits to State j in an interval of time $(0, t]$ given that it was in State 0 at time 0. We have

$$M_0(t) = \frac{1}{1 - H_{00}(t)}, \tag{3.18}$$

$$M_k(t) = \frac{u_1(t) * u_2(t) * \cdots * u_k(t)}{1 - H_{00}(t)}, \tag{3.19}$$

$$M_{D_0}(t) = \frac{G_0(t)}{1 - H_{00}(t)}, \tag{3.20}$$

$$M_{D_k}(t) = \frac{\{u_1(t) * u_2(t) * \cdots * u_k(t)\} * G_k(t)}{1 - H_{00}(t)}, \tag{3.21}$$

$$M_D(t) = \frac{\{u_1(t) * u_2(t) * \cdots * u_{N-1}\} * F_{N-1}(t)}{1 - H_{00}(t)}, \tag{3.22}$$

where $k = 1, 2, \cdots, N-1$. The LS transforms of these equations, $m_j(s)$, yields M_j as

$$\begin{aligned} M_j &= \lim_{t \to \infty} \frac{1}{t} M_j(t) \\ &= s \cdot m_j(s)|_{s=0}. \end{aligned} \tag{3.23}$$

Thus, we obtain

$$M_k = \frac{1}{l_{00}} \cdot \frac{1 - C_0(m)}{1 - C_k(m)}, \quad (k = 0, 1, \cdots, N-1), \tag{3.24}$$

$$M_{D_k} = \frac{1}{l_{00}} \cdot \frac{\{1 - C_0(m)\} C_k(m)}{1 - C_k(m)}, \quad (k = 0, 1, \cdots, N-1), \tag{3.25}$$

$$M_D = \frac{1 - C_0(m)}{l_{00}}. \tag{3.26}$$

We further derive the steady-state probabilities for State j, $P_j (j = 0, 1, \cdots, N-1, D_0, D_1, \cdots, D_{N-1}, D)$. Let $P_j(t)$ denote the transition probabilities that the process is in state j at time t given that it was in State 0 at time 0. We have

$$P_0(t) = \frac{1 - (F_0(t) + G_0(t))}{1 - H_{00}(t)}, \tag{3.27}$$

$$P_k(t) = \frac{\{u_1(t) * u_2(t) * \cdots * u_k(t)\} * \{1 - (F_k(t) + G_k(t))\}}{1 - H_{00}(t)}, \tag{3.28}$$

$$P_{D_0}(t) = \frac{G_0(t) * (1 - R_0(t))}{1 - H_{00}(t)}, \tag{3.29}$$

$$P_{D_k}(t) = \frac{\{u_1(t) * u_2(t) * \cdots * u_k(t)\} * G_k(t) * (1 - R_k(t))}{1 - H_{00}(t)}, \tag{3.30}$$

$$P_D(t) = \frac{\{u_1(t) * u_2(t) * \cdots * u_k(t)\} * F_{N-1}(t) * (1 - Z_N(t))}{1 - H_{00}(t)}, \tag{3.31}$$

where $k = 1, 2, \cdots, n-1$. Using the formula for the LS transforms of these equations

$$\begin{aligned} P_j &= \lim_{t \to \infty} P_j(t) \\ &= p_j(s)|_{s=0}, \end{aligned} \tag{3.32}$$

we can obtain

$$P_k = \frac{1}{l_{00}} \cdot \frac{\{1 - C_0(m)\}\{a_k + b_k \sum_{j=1}^{m} C_k(j)\}}{1 - C_k(m)}, \quad (k = 0, 1, \cdots, N-1), \tag{3.33}$$

$$P_{D_k} = \frac{1}{l_{00}} \cdot \frac{\{1 - C_0(m)\} C_k(m) \sigma_k}{1 - C_K(m)}, \quad (k = 0, 1, \cdots, N-1), \tag{3.34}$$

$$P_D = \frac{1}{l_{00}} \cdot \{1 - C_0(m)\} \phi_N. \tag{3.35}$$

Applying the above results, we can derive the availability A_v and the mean time between failures MTBF. From Equation (3.33), A_v is given by

$$A_v = \sum_{k=0}^{N-1} P_k$$

$$= \frac{\displaystyle\sum_{k=0}^{N-1} \frac{a_k + b_k \Sigma_{j=1}^{m} C_k(j)}{1 - C_k(m)}}{\displaystyle\sum_{k=0}^{N-1} \frac{a_k + b_k \Sigma_{j=1}^{m} C_k(j)}{1 - C_k(m)} + \sum_{k=0}^{N-1} \frac{C_k(m)\sigma_k}{1 - C_k(m)} + \phi_N}. \tag{3.36}$$

If MDT denote the mean down time, the formulae

$$A_v = \frac{\text{MTBF}}{\text{MTBF} + \text{MDT}}, \tag{3.37}$$

and

$$\text{MTBF} + \text{MDT} = \frac{1}{\displaystyle\sum_{k=0}^{N-1} M_{D_k} + M_D} \tag{3.38}$$

yield MTBF as follows:

$$\begin{aligned} \text{MTBF} &= A_v(\text{MTBF} + \text{MDT}) \\ &= \frac{\displaystyle\sum_{k=0}^{N-1} \frac{a_k + b_k \Sigma_{j=1}^{m} C_k(j)}{1 - C_k(m)}}{\displaystyle\sum_{k=0}^{N-1} \frac{C_k(m)}{1 - C_k(m)} + 1}. \end{aligned} \tag{3.39}$$

Let us next derive the availability in the transient state, $A_v(t)$, by applying a continuous-time Markov chain and the randomization technique [10] to the previous model. If we assume all the time distributions in the model to be exponential distributions, stochastic behavior of the system is described by a continuous-time Markov chain. Though a transient solution is analytically obtained for several continuous-time Markov chains, we numerically calculate the solution since the analytical derivation is quite difficult for the Markov chain with many states such as states in our model. The availability in the transient-state, $A_v(t)$, is derived from the transient-state probability vector obtained.

The state probability vector at time t for the continuous-time Markov chain with $2N+1$ states,

$$\pi(t) = (\pi_0(t), \cdots, \pi_{N-1}(t), \pi_{D_0}(t), \cdots, \pi_{D_{N-1}}(t), \pi_D(t)) \tag{3.40}$$

is transformed by the randomization technique as follows,

$$\pi(t) = \sum_{n=0}^{\infty} \frac{(\Lambda t)^n}{n!} e^{-\Lambda t} \phi(n), \tag{3.41}$$

where Λ is the maximum absolute value of diagonal elements of the infinitesimal generator $\boldsymbol{Q}$ for the continuous-time Markov chain, and $\phi(n)$ is the state probability vector for a discrete time Markov chain characterized by a transition probability matrix $\boldsymbol{P}$ which is transformed from the matrix $\boldsymbol{Q}$. Using the state probability vector, the availability at time t is given by

$$A_v(t) = \sum_{k=0}^{N-1} \pi_k. \tag{3.42}$$

3.4 Numerical Illustrations

In this section, let us first numerically evaluate the optimum number of successful retries, N_S, which maximizes the availability, A_v, the mean times between failures, MTBF, in the steady-state. We next calculate the availabilities, $A_v(t)$, in the transient-state for $N = N_S$ and N in the vicinity of N_S.

Assume that $r_{kj} = P^{k+j}$ $(k = 0, 1, \cdots, N-1; j = 1, 2, \cdots, m-1)$, where P is a initial probability with which a retry succeeds at first. the mean time incurred for removing a permanent failure and intermittent failure are assumed to be $\sigma_1 = \sigma_2 = \cdots = \sigma_{N-1} = 1$ [hour] and $\phi_N = \phi = 1, 5, 20, 50$ [hours], respectively. We apply a failure mode $a_k = 100/(k+1)$ [hours] $(k = 0, 1, \cdots, N-1)$ to the mean time between the successful retry and the failure. We further consider that an overhead for a retry is enough law to ignore, i.e., $b_k = 0$ [hour] $(k = 0, 1, \cdots, N-1)$.

Table 3.1 shows the optimum number of successful retries, N_S, which maximizes A_v, for $\phi = 1, P = 50, 60, 70, 80, 90, 99$ [%] and $m = 1, 2, 3, 4$. In the case of $\phi = 5, 20, 50$, the results are shown in Table 3.2, 3.3, 3.4, respectively. Moreover, the maximized availabilities corresponding to Table 3.1, ..., Table 3.4 are shown in Fig. 3.2, 3.3, 3.4, 3.5, respectively. Table 3.5 similarly shows the optimum number of successful retries, N_S, which maximizes MTBF, when the parameters are the same as in Table 3.1. The maximized MTBF is shown in Fig. 3.6. Note that MTBF is independent of σ_k and ϕ_N as is evident from Equation (3.39). From these results, for instance, we can see that in case $\phi = 1, m = 2$, and $P = 90$[%], the maximized availability 0.9939 is yielded at $N = 4$, and the maximized MTBF 164.5 [hours] is yielded at $N = 4$.

The number, N_S, which maximizes A_v increases when a initial probability, P, or a prescribed number of retries, m, increase as shown in Table 3.1

Table 3.1: The optimum number of successful retries maximizing the availability for $\phi = 1$.

m \ $P[\%]$	50	60	70	80	90	99
1	1	1	1	2	2	7
2	1	1	2	2	4	15
3	1	2	2	3	5	24
4	1	2	2	3	5	33

Table 3.2: The optimum number of successful retries maximizing the availability for $\phi = 5$.

m \ $P[\%]$	50	60	70	80	90	99
1	2	3	3	4	5	13
2	2	3	4	4	7	26
3	3	3	4	5	8	37
4	3	3	4	5	9	48

Table 3.3: The optimum number of successful retries maximizing the availability for $\phi = 20$.

m \ $P[\%]$	50	60	70	80	90	99
1	4	5	6	7	10	25
2	5	5	7	8	12	41
3	5	6	7	9	13	55
4	5	6	7	9	14	68

Table 3.4: The optimum number of successful retries maximizing the availability for $\phi = 50$.

m \ $P[\%]$	50	60	70	80	90	99
1	6	7	9	11	15	38
2	6	8	10	13	18	58
3	7	8	11	14	19	73
4	7	9	11	14	20	86

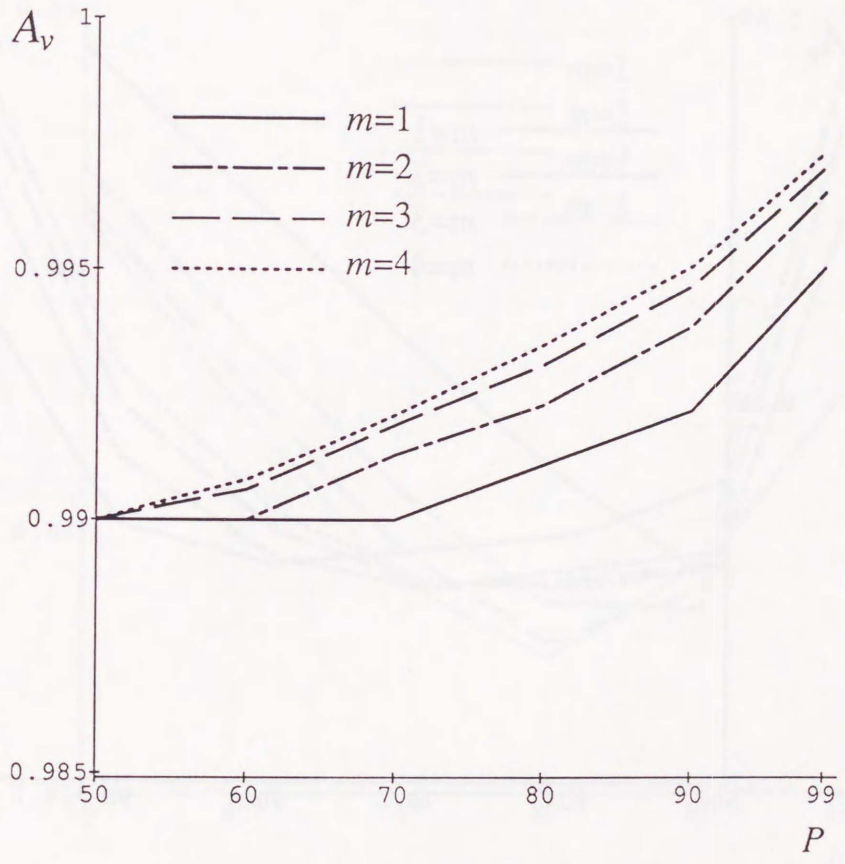

Figure 3.2: The maximized availability for $\phi = 1$.

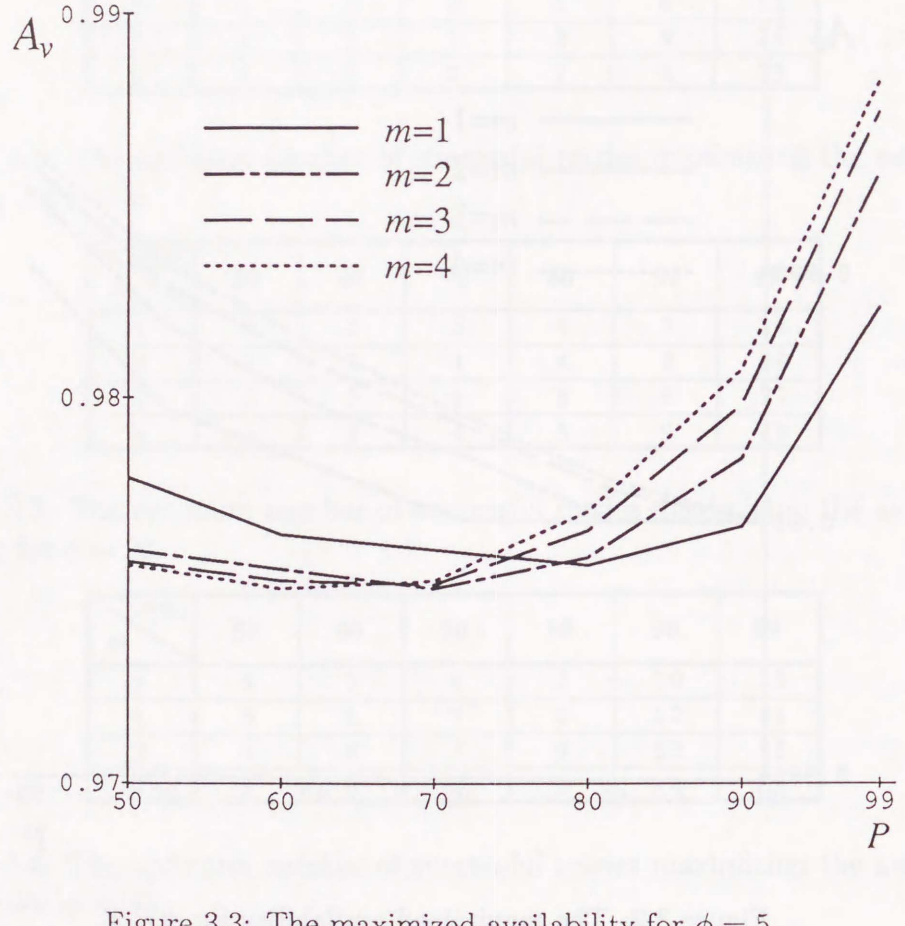

Figure 3.3: The maximized availability for $\phi = 5$.

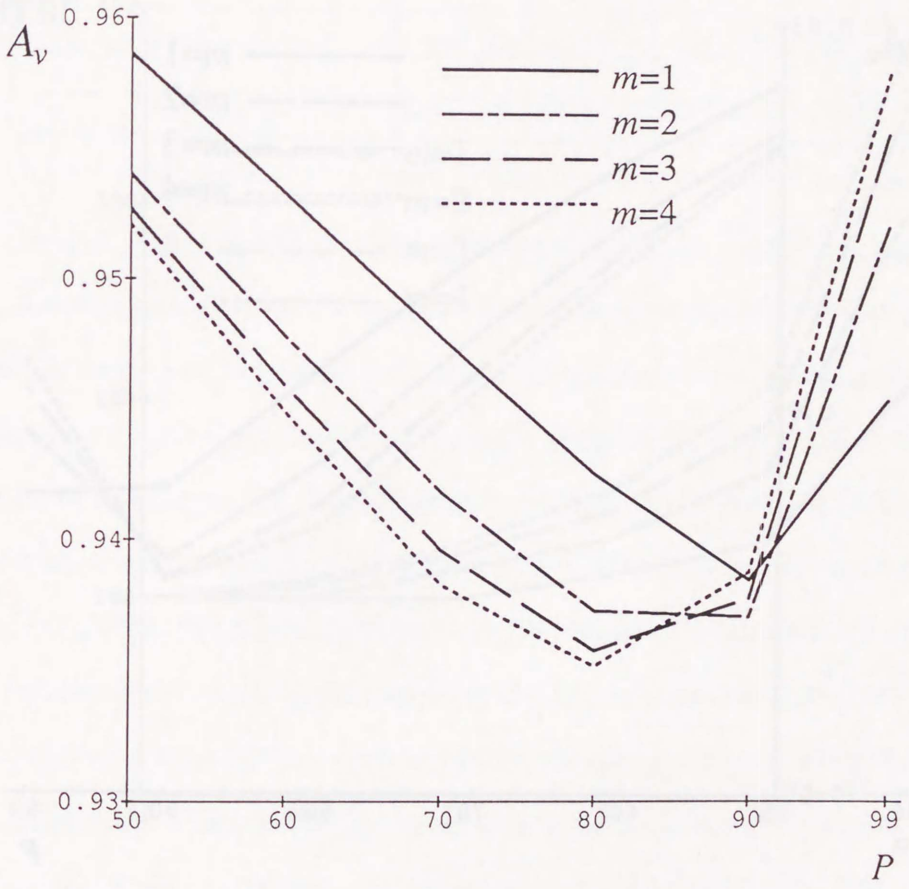

Figure 3.4: The maximized availability for $\phi = 20$.

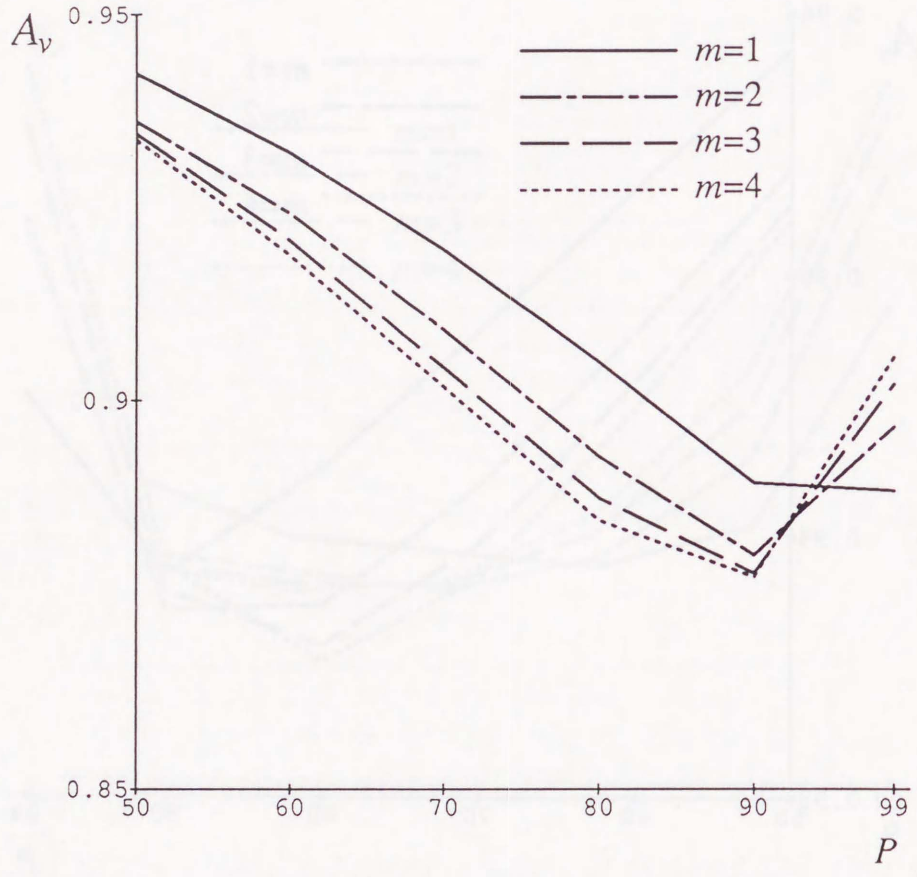

Figure 3.5: The maximized availability for $\phi = 50$.

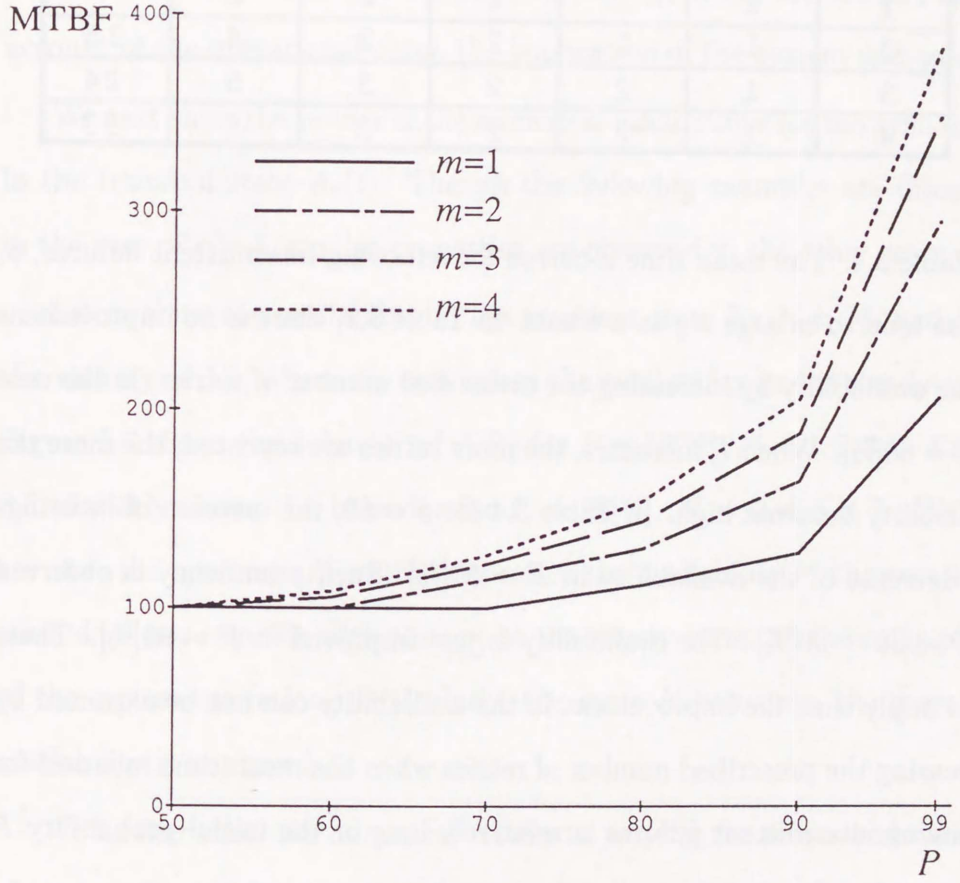

Figure 3.6: The maximized MTBF.

Table 3.5: The optimum number of successful retries maximizing MTBF.

m \ $P[\%]$	50	60	70	80	90	99
1	1	1	1	2	2	7
2	1	1	2	2	4	15
3	1	2	2	3	5	24
4	1	2	2	3	5	33

... Table 3.4. The mean time incurred for removing intermittent failures, ϕ, is also tend to enlarge N_S as a whole. In Table 3.1, there is no improvement in the availability by increasing the prescribed number of retries, in the case of $P = 50[\%]$. When P increases, the more retries are repeated, the more the availability becomes high. In Table 3.4 for $\phi = 50$, the increase of m brings the decrease of the availability in $P = 50[\%]$. Such a tendency is observed in $P = 50 \cdots 90[\%]$. The availability is just improved in $P = 99[\%]$. These facts imply that the improvement in the availability can not be expected by increasing the prescribed number of retries when the mean time incurred for removing intermittent failures is relatively long or the initial probability P is not high.

We can similarly see that the effects for MTBF by retries depends on the initial probability P, from Table 3.5 and Fig. 3.6. When P increases, N_S and MTBF increase, and these values also increase according as the prescribed number of retries increases.

The result of MTBF is independent of the overhead time for the maintenances as before. The optimum number of successful retries, N_S, for MTBF coincides with N_S for A_v in case $\phi = 1$. We should determine which N_S is adopted in the maintenance policy, N_S for A_v, or N_S for MTBF, taking account of the operational costs, the application of the system and so on.

We next show the results of the numerical calculations for the availability in the transient-state $A_v(t)$. Though the following examples are discussed in the case of $\phi = 5$, similar properties are observed in the other cases of ϕ.

Let us show the availability in the transient-state for $N = N_S$ and N in the vicinity of N_S, where N_S maximizes the availability in the steady-state. Figure 3.7 shows the behavior of $A_v(t)$ for $P = 50[\%]$, $N = 1, 2, 3, 4$. Figure 3.8 similarly shows the behavior for $P = 90[\%]$, $N = 3, 4, 5, 6, 7$. We can see that the optimum N which maximizes the availability in the transient-state does not coincide with the one in the steady-state. In the early phase of the system operation, particularly, the more N increases, the more the availability is improved.

We further show the numerical results in varying the prescribed number of retries. Figure 3.9 shows the behavior of the availability for $P = 50[\%]$, $m = 1, 2, 3, 4$, where we use N which maximizes the availability in the steady-state for each value of m. Figure 3.10 also shows the behavior for $P = 90[\%]$, where the other parameter is situated similarly to Fig. 3.9. In case $P = 50[\%]$, though the availability at $m = 1$ is the greatest in the steady-state,

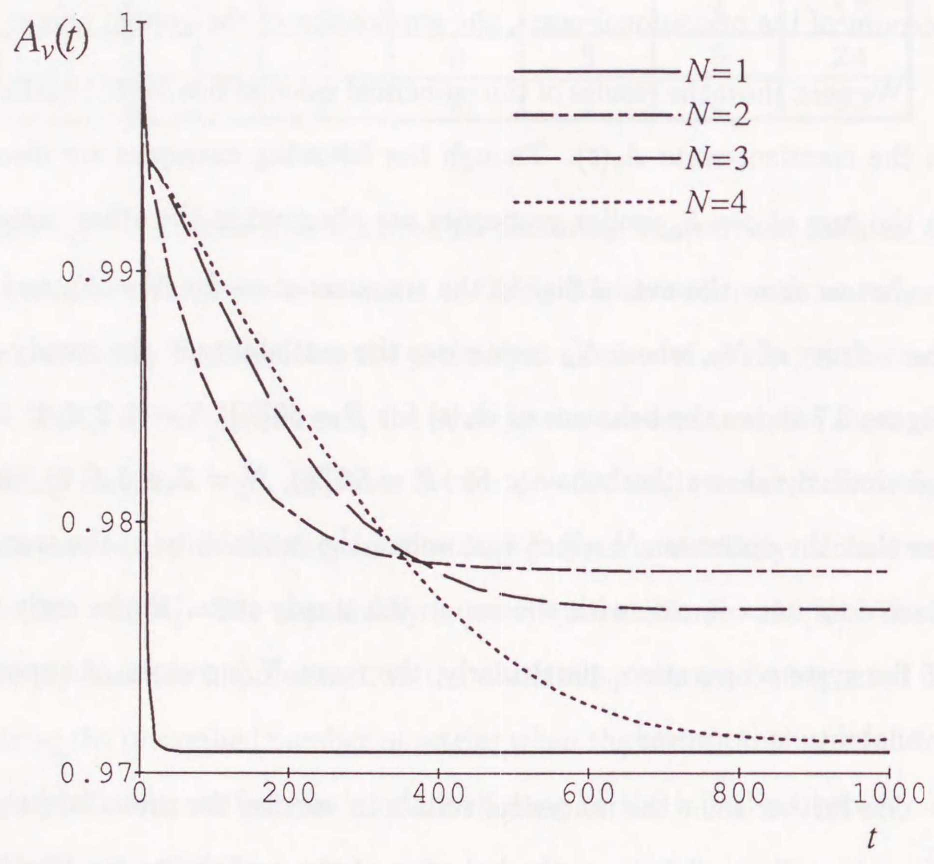

Figure 3.7: The behaviors of availability $A_v(t)$ for $P = 50$ in varying N ($\phi = 5, m = 1, N_S = 2$).

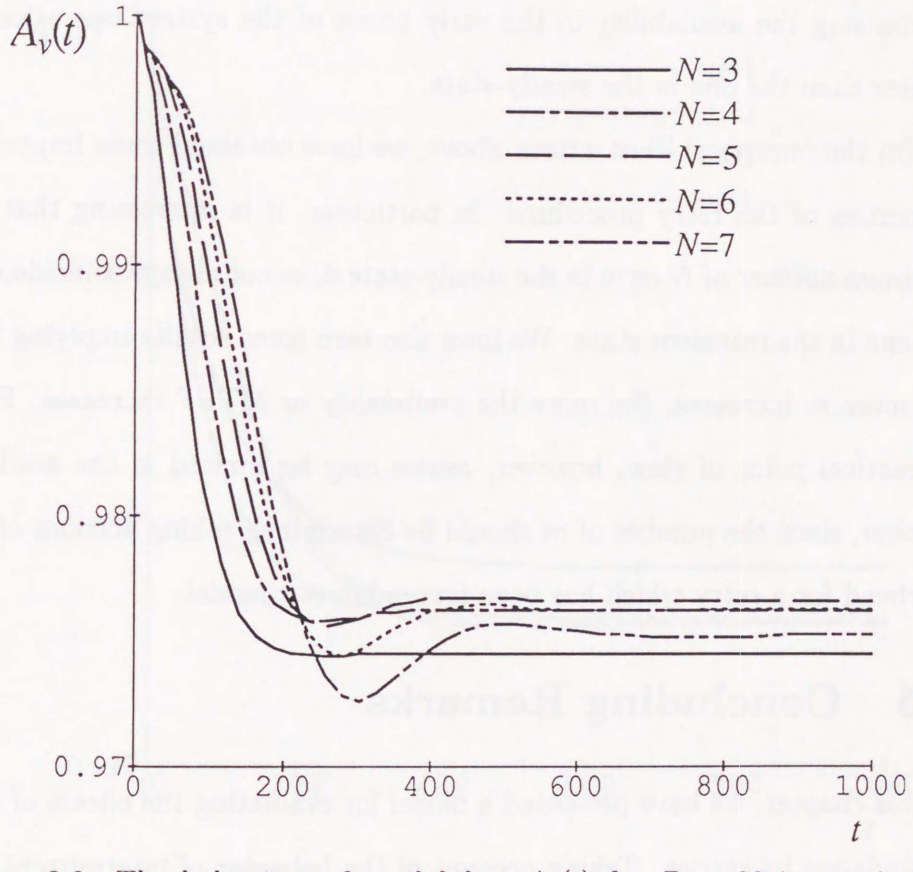

Figure 3.8: The behaviors of availability $A_v(t)$ for $P = 90$ in varying N ($\phi = 5, m = 1, N_S = 5$).

the availability at $m = 3$ is greater than the one at $m = 1$ for $0 \leq t \leq 180$. In case $P = 90[\%]$, the availability at $m = 4$ is greater than the others in the transient-state as well as the steady-state. These results imply that when the initial probability P is not high, the prescribed number of retries maximizing the availability in the early phase of the system operation, is greater than the one in the steady-state.

On the numerical illustrations above, we have obtained some important properties of the retry procedure. In particular, it is interesting that the optimum number of N or m in the steady-state does not always coincide with the one in the transient state. We have also seen some results implying that the more m increases, the more the availability or MTBF increases. From a practical point of view, however, retries may be limited in the available number, since the number of m should be determined taking account of the overhead for a retry which has been ignored in our model.

3.5 Concluding Remarks

In this chapter, we have presented a model for evaluating the effects of time redundancy by retries. Taking account of the behavior of intermittent and permanent failures, the system has been assumed to be maintained with the prescribed number of successful retries. The analysis of the model has yielded the reliability evaluation in the transient-state as well as the steady-state.

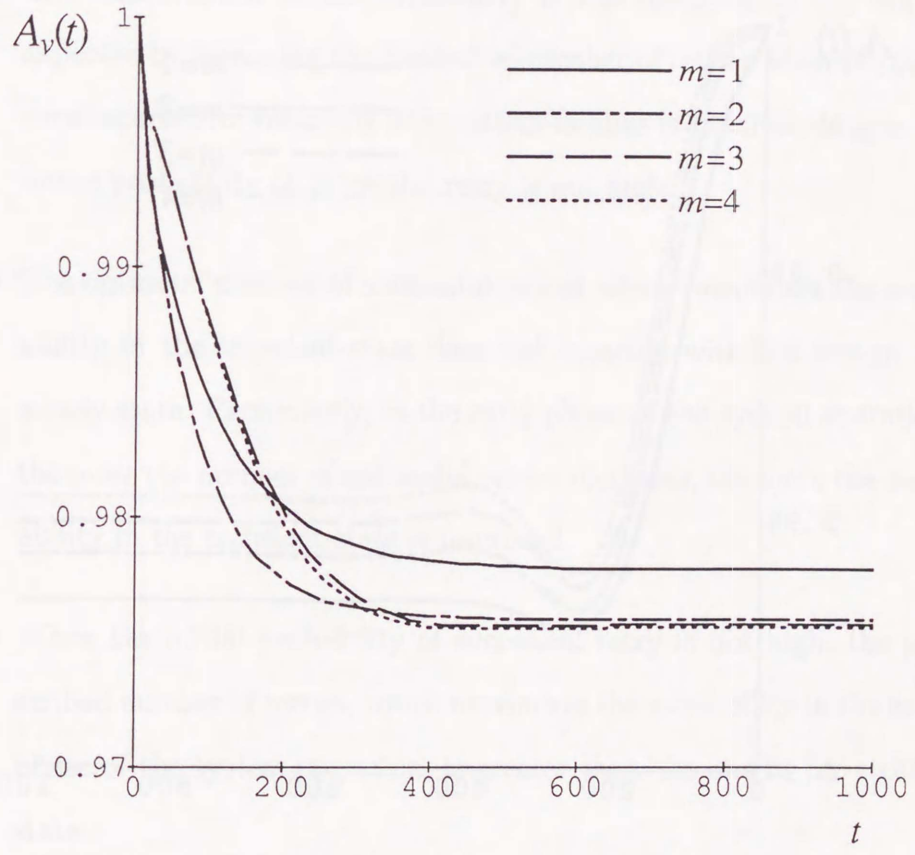

Figure 3.9: The behaviors of availability $A_v(t)$ for $P = 50$ in varying m ($\phi = 5$).

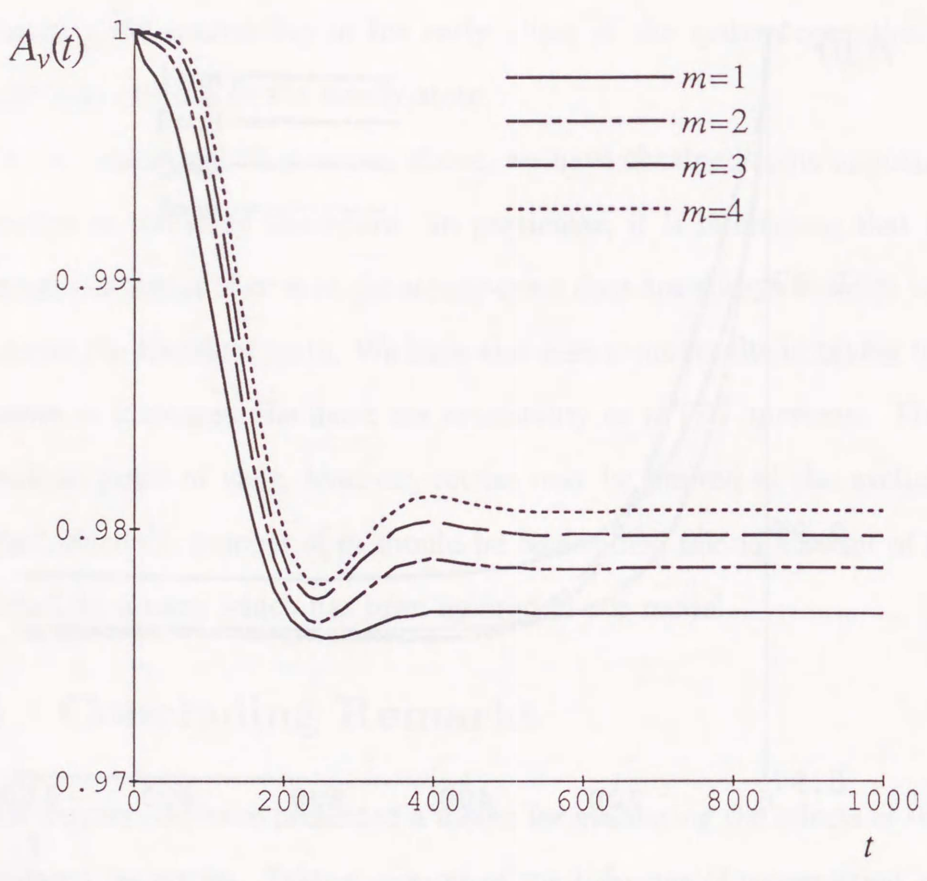

Figure 3.10: The behaviors of availability $A_v(t)$ for $P = 90$ in varying m ($\phi = 5$).

We have obtained several important properties of retry procedures from the analytical or numerical results. The principal results are shown as follows:

- The improvement in the availability in the steady-state can not be expected by increasing the prescribed number of retries when the mean time incurred for removing intermittent failures is relatively long or the initial probability of successful retry is not high.

- The optimum number of successful retries which maximizes the availability in the transient-state does not coincide with the one in the steady-state. Particularly, in the early phase of the system operation, the more the number of successful retries increases, the more the availability in the transient-state is improved.

- When the initial probability of successful retry is not high, the prescribed number of retries, which maximizes the availability in the early phase of the system operation, is greater than the one in the steady-state.

In the actual systems, a retry procedure is greatly useful for the component of a recovery mechanism as before. Specifying the parameters of our model statistically, we can roughly estimate the prescribed optimum number of retries or successful retries in many situations of the systems, and introduce the reasonable strategies for the retry procedures based on the results above.

References

[1] M. Mukaidono (ed.): *Introduction to Highly Reliable Techniques for Computer Systems*, Japanese Standards Association, Tokyo (1988) (in Japanese).

[2] H. Inose (ed.): *Reliable Computer Systems*, IPS Japan, Tokyo (1977) (in Japanese).

[3] K. Yasui, T. Nakagawa and Y. Sawa: "Reliability Analysis of a Computer System with Retry", *Trans. IEICE of Japan*, Vol. J64-D, No. 8, pp. 788-794 (1981) (in Japanese).

[4] S. Mallela and G. M. Masson: "Diagnosable Systems for Intermittent Faults", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 560-566 (1978).

[5] S. Y. H. Su, I. Koren and Y. K. Malaiya: "A Continuous-Parameter Markov Model and Detection Procedures for Intermittent Faults", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 567-570 (1978).

[6] Y. K. Malaiya: "Linearly Correlated Intermittent Failures", *IEEE Trans. Reliab.*, Vol. R-31, No. 2, pp. 211-215 (1982).

[7] K. Yasui and S. Osaki: "Optimum Inspection Policies for a Computer System with Intermittent Faults", *Trans. IPS Japan*, Vol. 30, No. 1, pp. 127-131 (1989) (in Japanese).

[8] T. Nakagawa and S. Osaki: "Markov Renewal Processes with Some Non-Regeneration Points and Their Applications to Reliability Theory", *Microelectron. Reliab.*, Vol. 15, pp. 633-636 (1976).

[9] S. M. Ross: *Stochastic Processes*, John Wiley & Sons, New York (1983).

[10] D. Gross and D. R. Miller: "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes", *Oper. Res.*, Vol. 32, No. 2, pp. 342-361 (1984).

Chapter 4

Reliability/Performance Evaluation for Multi-Processor Systems from the Viewpoint of Transaction Assignments

4.1 Introduction

The remarkable progress of modern computer technology enables us to make a large-scale and complex computing system which plays an important role in our society. Examples of such systems are telephone exchange systems, the on-line banking systems, vehicle control systems, seat reservation systems, and so on. A break-down of such systems may be costly, dangerous and may cause social panic. It is, therefore, of great importance to operate such computing systems with high reliability.

The fault-tolerant computing systems [1-4] have been proposed behind above background. The fault-tolerant computing systems are the systems which operate with high reliability using the technique of redundancy and/or maintenance. Several stochastic models for such systems have been proposed, and the reliability evaluations have been discussed [5-8]. For evaluating the fault-tolerant computing systems, the traditional reliability measures, such as the availability, the MTBF (Mean Time Between Failures), are not adequate since such systems assume not *up* and *down* state, but also several different levels. Therefore, we should propose new reliability and/or performance measures adequate to evaluate such systems. Several measures have been proposed [9-12].

Beaudry [9] proposed the performance-related reliability measures, such as the *computation availability* and the *MCBF (Mean Computation Between Failures)*, for such systems. Meyer [10] proposed the *performability* taking account of accomplishment levels from a user's viewpoint for a multi-processor system. Gay and Ketelsen [11] proposed the *throughput availability*, the *expected system throughput* and the *expected number of transactions lost* which are measures taking account of reliability, performance and computational demands simultaneously. Nakamura and Osaki [12] classified the lost jobs caused by processor failure and by cancellation (i.e., overflow of the arriving jobs). We will characterize the multi-processor systems using not only the traditional measures but also reliability/performance measures above to evaluate the systems.

In this chapter we analyze a multi-processor system which is one of the typical fault-tolerant computing systems, and is also called *A Gracefully Degrading System* from its redundant technique. Assuming that the system is composed of two processors and the buffer(s) (i.e., the storage facility of the arriving transactions). We propose two models from the viewpoint of transaction assignment, and obtain analytically the conventional reliability/performance measures for each model applying the Markov renewal processes [13, 14] and queueing theory [15]. We show numerical examples for evaluating two models from the viewpoint of transaction assignments. We finally show the impact of transaction assignment on the multi-processor systems.

4.2 Multi-Processor System Modeling

Consider a multi-processor system which is composed two processors and the buffer whose maximum storage capacity is $2N$. We assume that each processor obeys an exponential failure time distribution with failure rate λ_1 and an arbitrary repair time distribution $G_1(t)$ with mean repair time $1/\mu_1$. We assume that a buffer is composed of buffer elements, where each element obeys an exponential failure time distribution with rate λ_2 and an arbitrary repair time distribution $G_2(t)$ with mean time $1/\mu_2$. That is, if a buffer is composed of N elements, the exponential failure rate for the buffer is $N\lambda_2$ since it can be considered a series of buffer elements. There is a single repair facility and repair discipline is *first come, first served.* A processor or a buffer recovers its functioning upon repair completion.

Consider the behavior of the arriving transactions. The transactions arrive as a Poisson process with arrival rate λ_T and form a queue in the buffer. Each processor performs a transaction exponentially with processing rate μ_T. The break-down of the processor or the buffer implies that the existing transactions are lost. In particular, the break-down of a system implies that all transactions in the system are lost. When the system is under break-down or is occupied by the transactions of full capacity, the arriving transactions cannot be accepted (i.e., it is cancelled).

We propose the following two models from the viewpoint of transaction assignments:

Model 1: It is composed of two processors and a buffer with maximum capacity $2N$. The transactions are distributed to each processor after forming a queue in the buffer.

Model 2: It is composed of two processors and two buffers with maximum capacity N. That is, it is composed of two subsystems, where each subsystem is composed of a processor and a buffer. The arriving transactions are distributed to each subsystem with even probability.

Figure 4.1 shows a configuration of each model. If two processors break down simultaneously or a buffer breaks down, the system breaks down in Model 1. If a processor or a buffer breaks down, the subsystem breaks down, and if two subsystems break down simultaneously, the system breaks down in Model 2. The discipline of transaction assignment is uniform for Model 1, and is random for Model 2, respectively [11].

4.3 Model 1

Applying Markov renewal processes and queueing theory, we analyze the above Models. Nakamura and Osaki [12] analysed a multi-processor system by considering both the behaviors of the processors and of the queue of the transactions. They derived the exact formulae and approximate formulae by considering the behaviors to be *simultaneous* and *independent*, respectively, and showed that the approximate formulae are sufficiently precise. We apply this result in our models. That is, we analyse our models by considering both the behaviors of the processors and of the queue to be independent.

To analyze Model 1, we first define the following states which characterize the behavior of the system:

State 0: All units are operating.

State 1: Through state 0, one of the processors breaks down and its repair starts.

State 2: Through state 1, the remaining processor breaks down (system break-down).

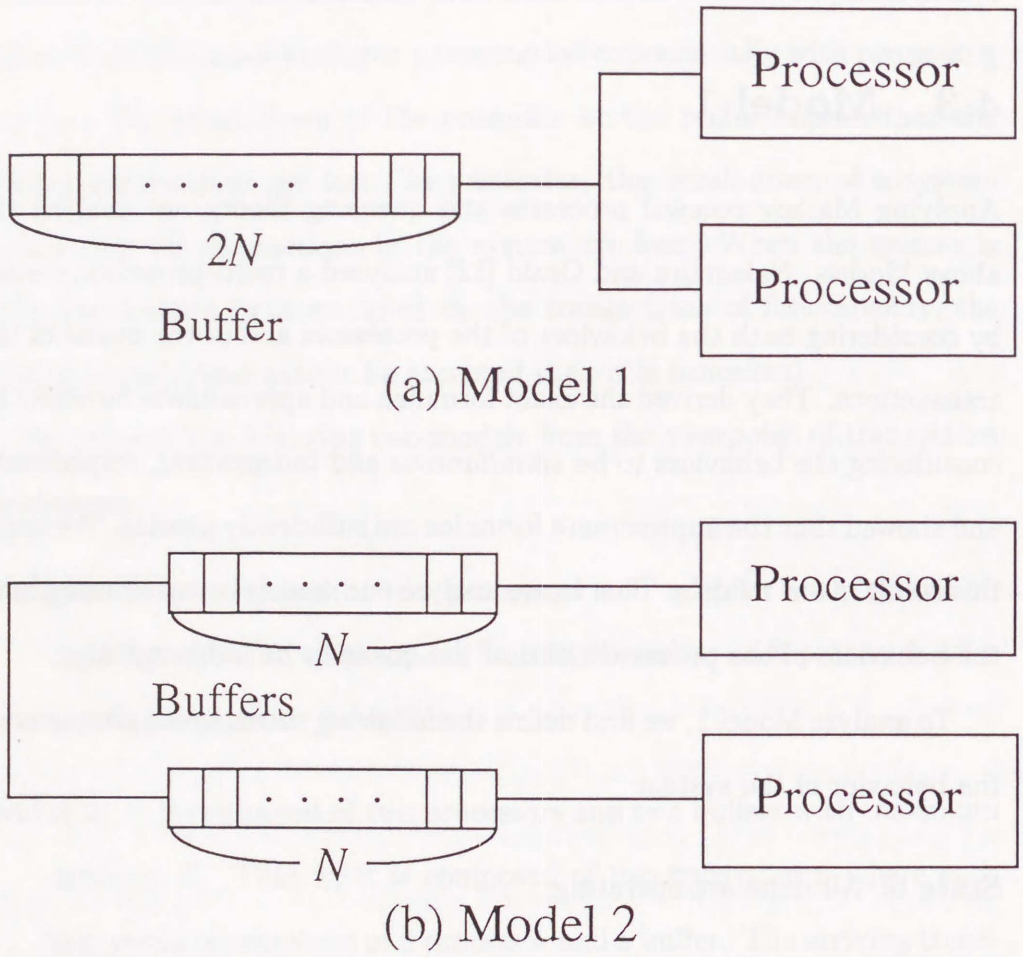

Figure 4.1: The configuration of the multi-processor system.

State 3: Through state 1, the buffer breaks down (system break-down).

State 4: Through state 0, the buffer breaks down (system break-down).

The state transition diagram among the states above is shown in Fig. 4.2, where the number *circled* denotes a regeneration point and the number *squared* denotes a non-regeneration point [14].

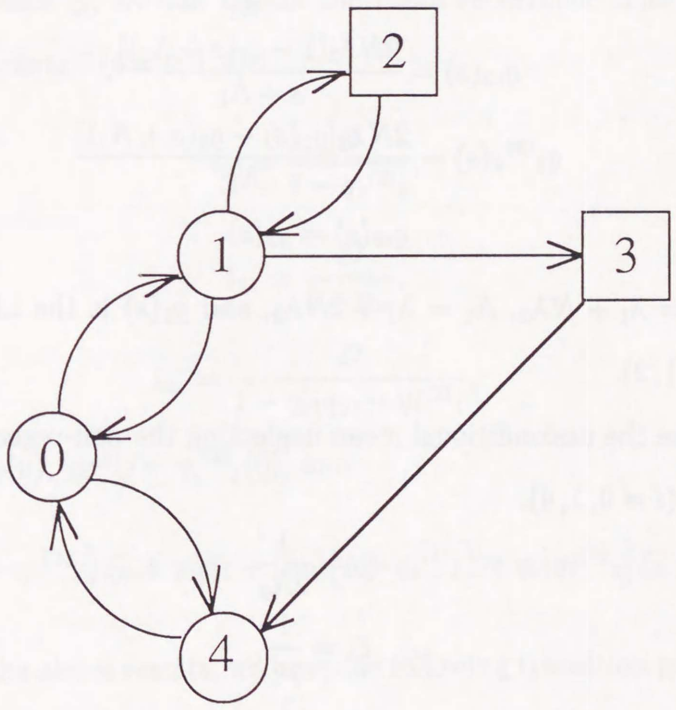

Figure 4.2: The state transition diagram for Model 1.

Let $q_{ij}(s)$ and $q_i{}^{(k)}{}_j(s)$ be the LS (Laplace-Stieltjes) transforms of one-step and two-step transition probabilities $Q_{ij}(t)$ and $Q_i{}^{(k)}{}_j(t)$, respectively.

Then we have

$$q_{01}(s) = \frac{2\lambda_1}{s+2\Lambda_0}, \tag{4.1}$$

$$q_{04}(s) = \frac{2N\lambda_2}{s+2\Lambda_0} \tag{4.2}$$

$$q_{10}(s) = g_1(s+\Lambda_1), \tag{4.3}$$

$$q_{12}(s) = \frac{\lambda_1[1-g_1(s+\Lambda_1)]}{s+\Lambda_1}, \tag{4.4}$$

$$q_1{}^{(2)}{}_1(s) = \frac{\lambda_1[g_1(s)-g_1(s+\Lambda_1)]}{\Lambda_1}, \tag{4.5}$$

$$q_{13}(s) = \frac{2N\lambda_2[1-g_1(s+\Lambda_1)]}{s+\Lambda_1}, \tag{4.6}$$

$$q_1{}^{(3)}{}_4(s) = \frac{2N\lambda_2[g_1(s)-g_1(s+\Lambda_1)]}{\Lambda_1}, \tag{4.7}$$

$$q_{40}(s) = g_2(s), \tag{4.8}$$

where $\Lambda_0 = \lambda_1 + N\lambda_2$, $\Lambda_1 = \lambda_1 + 2N\lambda_2$, and $g_1(s)$ is the LS transform of $G_i(t)$ $(i = 1, 2)$.

Let ξ_i be the unconditional mean neglecting the non-regeneration points for state i $(i = 0, 1, 4)$:

$$\xi_0 = \frac{1}{2\Lambda_0}, \tag{4.9}$$

$$\xi_1 = \frac{1}{\mu_1}, \tag{4.10}$$

$$\xi_4 = \frac{1}{\mu_2}, \tag{4.11}$$

and η_i be the unconditional mean *not* neglecting the non-regeneration points for state i $(i = 0, 1, 2, 3, 4)$:

$$\eta_i = \xi_i \qquad (i = 0, 4), \tag{4.12}$$

$$\eta_1 = \frac{1 - g_1(\Lambda_1)}{\Lambda_1}, \tag{4.13}$$

$$\eta_2 = \frac{\lambda_1 \left[\frac{1}{\mu_1} - \frac{1-g_1(\Lambda_1)}{\Lambda_1}\right]}{\Lambda_1}, \tag{4.14}$$

$$\eta_3 = \frac{2N\lambda_2 \left[\frac{1}{\mu_1} - \frac{1-g_1(\Lambda_1)}{\Lambda_1}\right]}{\Lambda_1}. \tag{4.15}$$

Using the limiting transition probabilities $q_{ij}(0)$, $q_i{}^{(k)}{}_j(0)$ and the unconditional means ξ_i, we can obtain the mean recurrence time for all the regeneration points i $(i = 0, 1, 4)$:

$$l_{00} = \frac{D}{1 - q_1{}^{(2)}{}_1}, \tag{4.16}$$

$$l_{11} = \frac{D}{1 - q_{04}}, \tag{4.17}$$

$$l_{44} = \frac{D}{1 - q_{01}q_{10} - q_1{}^{(2)}{}_1}, \tag{4.18}$$

where $q_{ij} = q_{ij}(0)$, $q_i{}^{(k)}j = q_i{}^{(k)}{}_j(0)$, and

$$D = \left(1 - q_1{}^{(2)}{}_1\right)\xi_0 + q_{01}\xi_1 + \left[q_{04}\left(1 - q_1{}^{(2)}{}_1\right) + q_{01}q_1{}^{(3)}{}_4\right]\xi_4. \tag{4.19}$$

Applying the above results, we have the following transition probabilities for all state i $(i = 0, 1, 2, 3, 4)$ in the steady-state:

$$P_i = \frac{\eta_i}{l_{ii}} \qquad (i = 0, 1, 4), \tag{4.20}$$

$$P_i = \frac{\eta_i}{l_{11}} \qquad (i = 2, 3). \tag{4.21}$$

We also have the following expected numbers of visits to all state i ($i = 0, 1, 2, 3, 4$) per unit time in the steady-state:

$$M_i = \frac{1}{l_{ii}} \qquad (i = 0, 1, 4), \tag{4.22}$$

$$M_i = \frac{q_{1i}}{l_{11}} \qquad (i = 2, 3). \tag{4.23}$$

Let us next consider the behavior of the arriving transactions. The transactions form an M/M/2/$2N$+2 queue in state 0. Let $P_j^{(0)}$ denote the steady-state probability that the number of transactions in system is j in state 0. Then $P_j^{(0)}$ ($j = 0, 1, \cdots, 2N + 2$) is given by

$$P_0^{(0)} = \frac{1}{1 + \frac{\lambda_T}{\mu_T} + \frac{1}{2}\left(\frac{\lambda_T}{\mu_T}\right)^2 + 2\sum_{k=3}^{2N+2}\left(\frac{\lambda_T}{2\mu_T}\right)^k}, \tag{4.24}$$

$$P_1^{(0)} = \left(\frac{\lambda_T}{\mu_T}\right) P_0^{(0)}, \tag{4.25}$$

$$P_j^{(0)} = 2\left(\frac{\lambda_T}{2\mu_T}\right)^j P_0^{(0)} \qquad (j = 1, 2, \cdots, 2N + 2). \tag{4.26}$$

The transactions form an M/M/1/$2N + 1$ queue in state 1. Let $P_j^{(1)}$ denote the steady-state probability that the number of transactions in the system is j in state 1. Then we have

$$P_j^{(1)} = \frac{(1 - \rho_1)\rho_1^j}{1 - \rho_1^{2N+2}} \qquad (\rho_1 \neq 1), \tag{4.27}$$

$$P_j^{(1)} = \frac{1}{2N + 2} \qquad (\rho_1 = 1), \tag{4.28}$$

where $\rho_1 = \lambda_T/\mu_T$ and $j = 0, 1, \cdots, 2N + 1$.

Applying the results in Equations (4.20)-(4.28), we can obtain the following reliability/performance measures:

(i) The steady-state availability A_V:

$$A_V = P_0 + P_1. \tag{4.29}$$

(ii) MTBF (Mean Time Between Failures):

$$\mathrm{MTBF} = \frac{A_V}{M_2 + M_3 + M_4}. \tag{4.30}$$

(iii) The computation availability A_C [9]:

$$A_C = 2\mu_T P_0 + \mu_T P_1. \tag{4.31}$$

(iv) The expected system throughput T_P [11]:

$$T_P = 2\mu_T \left(1 - P_0^{(0)} - \frac{P_1^{(0)}}{2}\right) P_0 + \mu_T \left(1 - P_0^{(1)}\right) P_1. \tag{4.32}$$

(v) The expected number of lost jobs by cancellation per unit time in the steady-state, C_j [12]:

$$C_J = \lambda_T \left(P_0 P_{2N+2}^{(0)} + P_1 P_{2N+1}^{(1)} + P_2 + P_3 + P_4\right). \tag{4.33}$$

Note that the following identity holds:

$$T_P + C_J = \lambda_T. \tag{4.34}$$

4.4 Model 2

Just similar to Model 1, we define the following states for Model 2:

State 0: All units are operating.

State 1: Through state 0, one of the processors breaks down and its repair starts. The system is operating as degrading the remaining subsystem function.

State 2: Through state 1, the remaining processor breaks down (system break-down).

State 3: Through state 1, the operating buffer breaks down (system break-down).

State 4: Through state 0, one of the buffers breaks down and its repair starts. The system is operating as degrading the remaining subsystem function.

State 5: Through state 4, the remaining buffer breaks down (system break-down).

State 6: Through state 4, the operating processor breaks down (system break-down).

The state transition diagram among the states above is shown in Fig. 4.3.

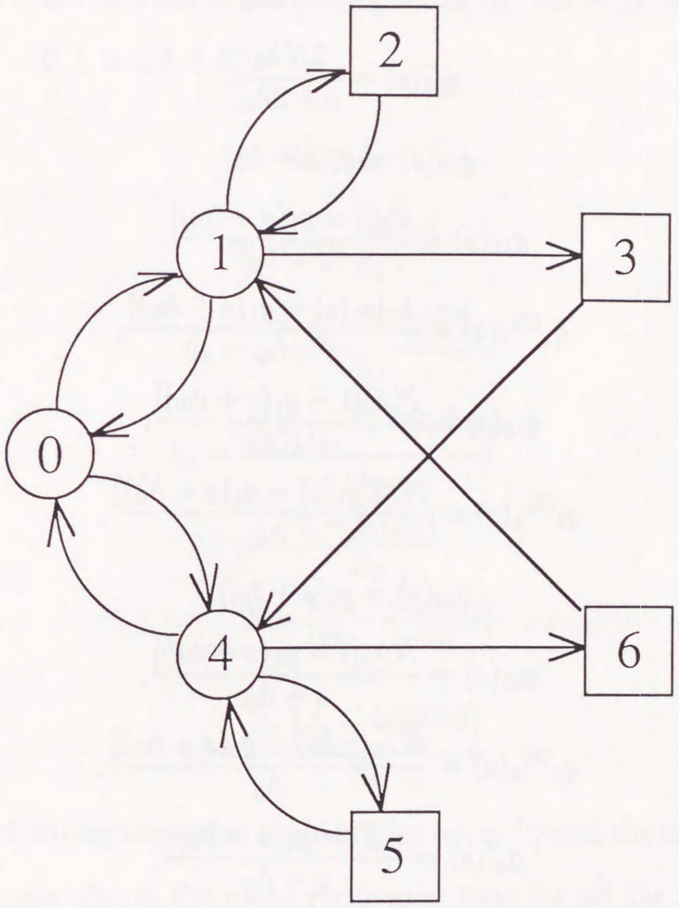

Figure 4.3: The state transition diagram for Model 2.

Let $q_{ij}(s)$ and $q_i{}^{(k)}{}_j(s)$ be the LS transforms one-step and two-step transition probabilities $Q_{ij}(t)$ and $Q_i{}^{(k)}{}_j(t)$, respectively. Then we have

$$q_{01}(s) = \frac{2\lambda_1}{s + 2\Lambda_0}, \tag{4.35}$$

$$q_{04}(s) = \frac{2N\lambda_2}{s + 2\Lambda_0}, \tag{4.36}$$

$$q_{10}(s) = g_1(s + \Lambda_0), \tag{4.37}$$

$$q_{12}(s) = \frac{\lambda_1[1 - g_1(s + \Lambda_0)]}{s + \Lambda_0}, \tag{4.38}$$

$$q_1{}^{(2)}{}_1(s) = \frac{\lambda_1[g_1(s) - g_1(s + \Lambda_0)]}{\Lambda_0}, \tag{4.39}$$

$$q_{13}(s) = \frac{N\lambda_2[1 - g_1(s + \Lambda_0)]}{s + \Lambda_0}, \tag{4.40}$$

$$q_1{}^{(3)}{}_4(s) = \frac{N\lambda_2[g_1(s) - g_1(s + \Lambda_0)]}{\Lambda_0}, \tag{4.41}$$

$$q_{40}(s) = g_2(s + \Lambda_0), \tag{4.42}$$

$$q_{45}(s) = \frac{N\lambda_2[1 - g_2(s + \Lambda_0)]}{s + \Lambda_0}, \tag{4.43}$$

$$q_4{}^{(5)}{}_4(s) = \frac{N\lambda_2[g_2(s) - g_2(s + \Lambda_0)]}{\Lambda_0}, \tag{4.44}$$

$$q_{46}(s) = \frac{\lambda_1[1 - g_2(s + \Lambda_0)]}{s + \Lambda_0}, \tag{4.45}$$

$$q_4{}^{(6)}{}_1(s) = \frac{\lambda_1[g_2(s) - g_2(s + \Lambda_0)]}{\Lambda_0}. \tag{4.46}$$

Let ξ_i be the unconditional mean neglecting the non-regeneration points for state i ($i = 0, 1, 4$):

$$\xi_0 = \frac{1}{2\Lambda_0}, \tag{4.47}$$

$$\xi_1 = \frac{1}{\mu_1}, \tag{4.48}$$

$$\xi_4 = \frac{1}{\mu_2}, \tag{4.49}$$

and η_i be the unconditional mean *not* neglecting the non-regeneration points for state i $(i = 0, 1, 2, 3, 4, 5, 6)$:

$$\eta_0 = \xi_0, \tag{4.50}$$

$$\eta_1 = \frac{1 - g_1(\Lambda_0)}{\Lambda_0}, \tag{4.51}$$

$$\eta_2 = \frac{\lambda_1 \left[\frac{1}{\mu_1} - \frac{1-g_1(\Lambda_0)}{\Lambda_0}\right]}{\Lambda_0}, \tag{4.52}$$

$$\eta_3 = \frac{N\lambda_2 \left[\frac{1}{\mu_1} - \frac{1-g_1(\Lambda_0)}{\Lambda_0}\right]}{\Lambda_0}. \tag{4.53}$$

$$\eta_4 = \frac{1 - g_2(\Lambda_0)}{\Lambda_0}, \tag{4.54}$$

$$\eta_5 = \frac{N\lambda_2 \left[\frac{1}{\mu_2} - \frac{1-g_2(\Lambda_0)}{\Lambda_0}\right]}{\Lambda_0}. \tag{4.55}$$

$$\eta_6 = \frac{\lambda_1 \left[\frac{1}{\mu_2} - \frac{1-g_2(\Lambda_0)}{\Lambda_0}\right]}{\Lambda_0}. \tag{4.56}$$

Using the limiting transition probabilities q_{ij}, ${q_i}^{(k)}{}_j$ and the unconditional means ξ_i, we can obtain the mean recurrence time for all the regeneration point i $(i = 0, 1, 4)$:

$$l_{00} = \frac{D}{1 - {q_1}^{(3)}{}_4 {q_4}^{(6)}{}_1 - {q_1}^{(2)}{}_1 - {q_4}^{(5)}{}_4 + {q_1}^{(2)}{}_1 {q_4}^{(5)}{}_4}, \tag{4.57}$$

$$l_{11} = \frac{D}{1 - q_{04}q_{40} - {q_4}^{(5)}{}_4}, \tag{4.58}$$

$$l_{44} = \frac{D}{1 - q_{01}q_{10} - q_1{}^{(2)}{}_1}, \tag{4.59}$$

where

$$D = \left(1 - q_1{}^{(3)}{}_4 q_4{}^{(6)}{}_1 - q_1{}^{(2)}{}_1 - q_4{}^{(5)}{}_4 + q_1{}^{(2)}{}_1 q_4{}^{(5)}{}_4\right)\xi_0$$

$$+ \left[q_{01}\left(1 - q_4{}^{(5)}{}_4\right) + q_{04}q_4{}^{(6)}{}_1\right]\xi_1 + \left[q_{04}\left(1 - q_1{}^{(2)}{}_1\right) + q_{01}q_1{}^{(3)}{}_4\right]\xi_4. \tag{4.60}$$

Applying the above results, we have the following transition probability for state i $(i = 0, 1, \cdots, 6)$ in the steady-state:

$$P_i = \frac{\eta_i}{l_{ii}} \qquad (i = 0, 1, 4), \tag{4.61}$$

$$P_i = \frac{\eta_i}{l_{11}} \qquad (i = 2, 3), \tag{4.62}$$

$$P_i = \frac{\eta_i}{l_{44}} \qquad (i = 5, 6). \tag{4.63}$$

We also have the following expected number of visits to state i $(i = 0, 1, \cdots, 6)$ per unit time in the steady-state:

$$M_i = \frac{1}{l_{ii}} \qquad (i = 0, 1, 4), \tag{4.64}$$

$$M_i = \frac{q_{1i}}{l_{11}} \qquad (i = 2, 3), \tag{4.65}$$

$$M_i = \frac{q_{4i}}{l_{44}} \qquad (i = 5, 6). \tag{4.66}$$

Let us next consider the behavior of the arriving transactions. The transactions form an M/M/1/$N+1$ queue with arrival rate $\lambda_T/2$ in each subsystem in state 0. Let $P_j^{(0)}$ denote the steady-state probability that the number of transactions in the subsystem is j in state 0. Then we have

$$P_j^{(0)} = \frac{(1-\rho_2)\rho_2^j}{1-\rho_2^{N+2}} \quad (\rho_2 \neq 1), \tag{4.67}$$

$$P_j^{(0)} = \frac{1}{N+2} \quad (\rho_2 = 1), \tag{4.68}$$

where $\rho_2 = \lambda_T/2\mu_T$ and $j = 0, 1, \cdots, N+1$. The transactions form an M/m/1/$N+1$ queue with arrival rate λ_T in the remaining subsystem in state 1 or 4. Let $P_j^{(1)}$ denote the steady-state probability that the number of transactions in the subsystem is j in state 1 or 4. Then we have

$$P_j^{(1)} = \frac{(1-\rho_1)\rho_1^j}{1-\rho_1^{N+2}} \quad (\rho_1 \neq 1), \tag{4.69}$$

$$P_j^{(1)} = \frac{1}{N+2} \quad (\rho_1 = 1), \tag{4.70}$$

where $j = 0, 1, \cdots, N+1$.

Applying the results in Equations (4.61)-(4.70), we can obtain the following reliability/performance measures:

(i) The steady-state availability A_V:

$$A_V = P_0 + P_1 + P_4. \tag{4.71}$$

(ii) MTBF:

$$\text{MTBF} = \frac{A_V}{M_2 + M_3 + M_5 + M_6}. \tag{4.72}$$

(iii) The computation availability A_C:

$$A_C = 2\mu_T P_0 + \mu_T(P_1 + P_4). \tag{4.73}$$

(iv) The expected system throughput T_P:

$$T_P = 2\mu_T \left(1 - P_0^{(0)}\right) P_0 + \mu_T \left(1 - P_0^{(1)}\right) (P_1 + P_4). \tag{4.74}$$

(v) The expected number of lost jobs by cancellation per unit time in the steady-state, C_J:

$$C_J = \lambda_T \left[P_0 P_{N+1}^{(0)} + (P_1 + P_4) P_{N+1}^{(1)} + P_2 + P_3 + P_5 + P_6\right]. \tag{4.75}$$

Note that the following identity holds:

$$T_P + C_J = \lambda_T. \tag{4.76}$$

4.5 Numerical Illustrations

Let us numerically compute the reliability/performance measures obtained in the preceding sections. Applying the appropriate values to all the parameters, we can derive the following results of the numerical examples. These results hold even in the case where the parameters vary in the ordinary range. Assume that the arbitrary repair time distribution is a gamma distribution of order 2:

$$G_i = 1 - (1 + 2\mu_i t) exp(-2\mu_i t) \quad (i = 1, 2). \tag{4.77}$$

Let us discuss a case in which the buffers cannot break-down, i.e., $\lambda_2 = 0$. In this case, the stochastic behaviors of each model are identical since the behavior of the system is just the behavior of the two processors. Note that A_V, MTBF and A_C are entirely identical for each model, respectively. Figure 4.4 shows the dependence of N on C_J for each model, where $\lambda_1 = 0.001$, $\mu_1 = 1$, $\lambda_T = 15000$ and $\mu_T = 10000$. The more N increases, the more C_J decreases for each model, and C_J of Model 1 always is less than that of Model 2. From this numerical example, the uniform assignment (i.e., Model 1) is superior to the random assignment (i.e., Model 2) as Gay and Ketelsen showed [11]. Note that the behaviors of C_J and T_P are symmetrical, since identity Equation (4.34) or (4.76) hold. Hence, the two measures are equivalent in the evaluation.

Let us next discuss a case in which the buffers can break down, i.e., $\lambda_2 > 0$. In this case, the stochastic behaviors of each model are not identical since the configurations of each model are different. Figure 4.5 shows the values of A_V in varying N for each model, where all the parameters are specified just same as in Fig. 4.4 and $\lambda_2 = 0.0001$, $\mu_2 = 1$. When N increases, A_V of each model decreases and, in particular, A_V of Model 2 is always greater than that of Model 1. We can obtain a similar result with respect to MTBF. Figure 4.6 shows the values of A_C in varying N for each model, where all the parameters are specified just same as in Fig. 4.5. The increase in buffers causes the decrease in A_C. When N increases, the decrease in A_C of Model 2 is always less than that of Model 1. These facts show that Model 2 is superior to Model 1 from the viewpoint of *fault-tolerance*.

Figure 4.7 shows the dependence of N on C_J for each model, where all

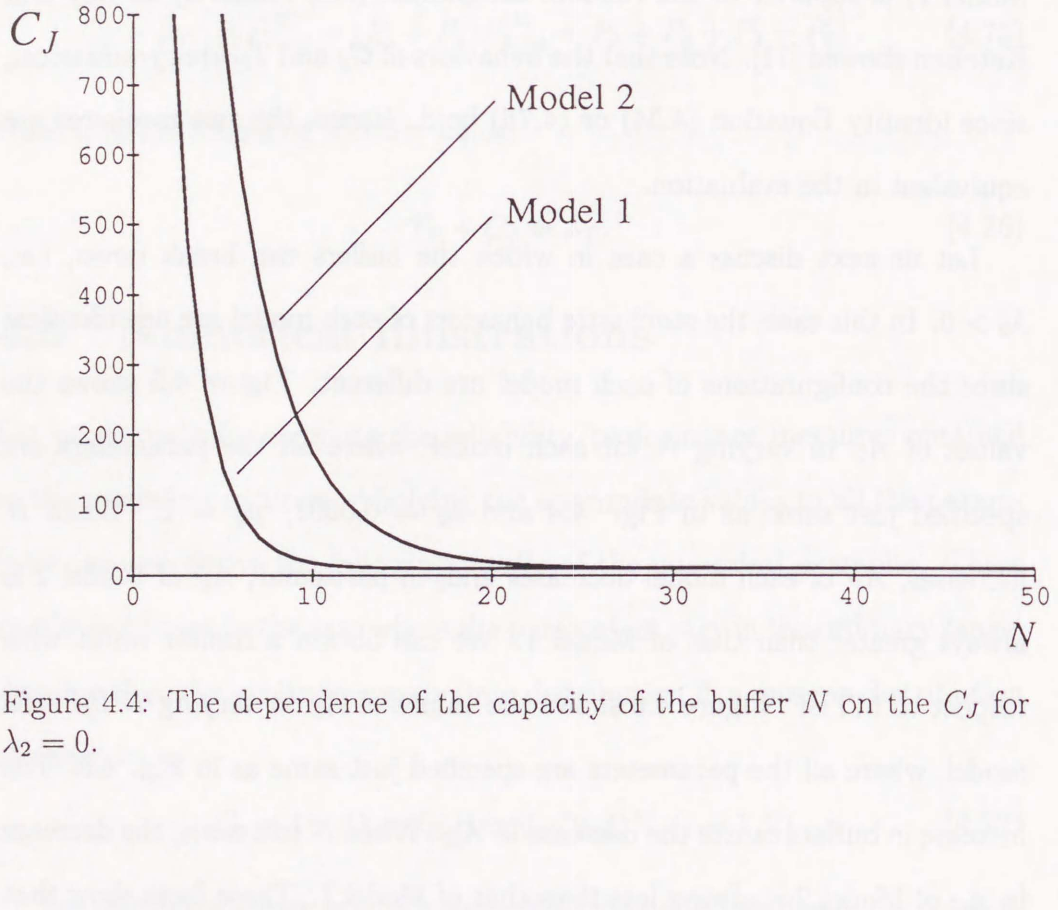

Figure 4.4: The dependence of the capacity of the buffer N on the C_J for $\lambda_2 = 0$.

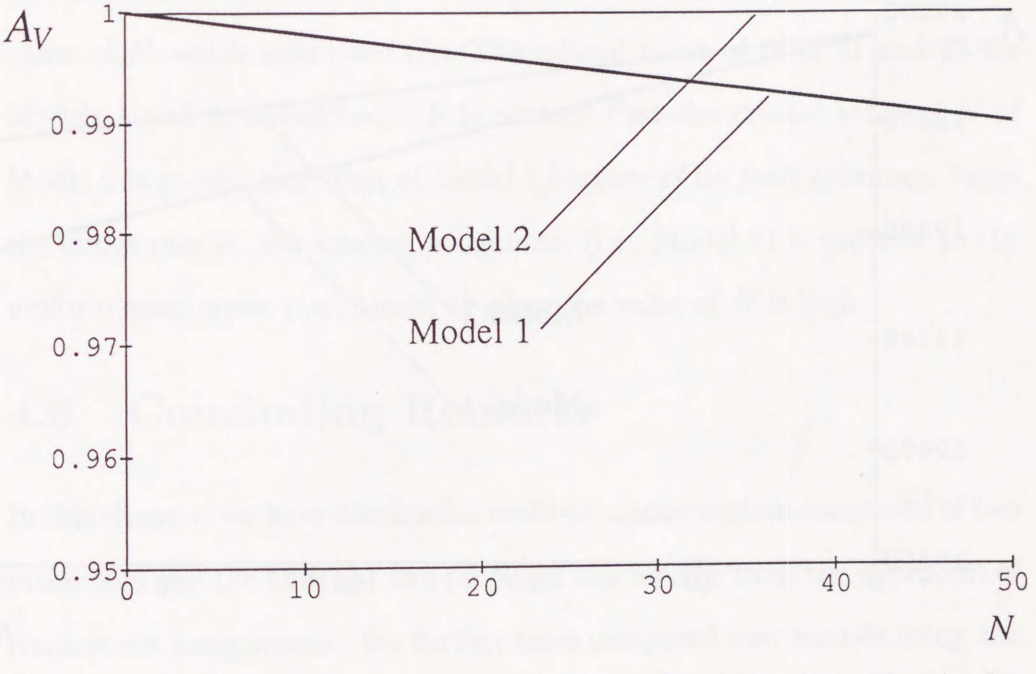

Figure 4.5: The dependence of the capacity of the buffer N on the A_V for $\lambda_2 = 0.0001$.

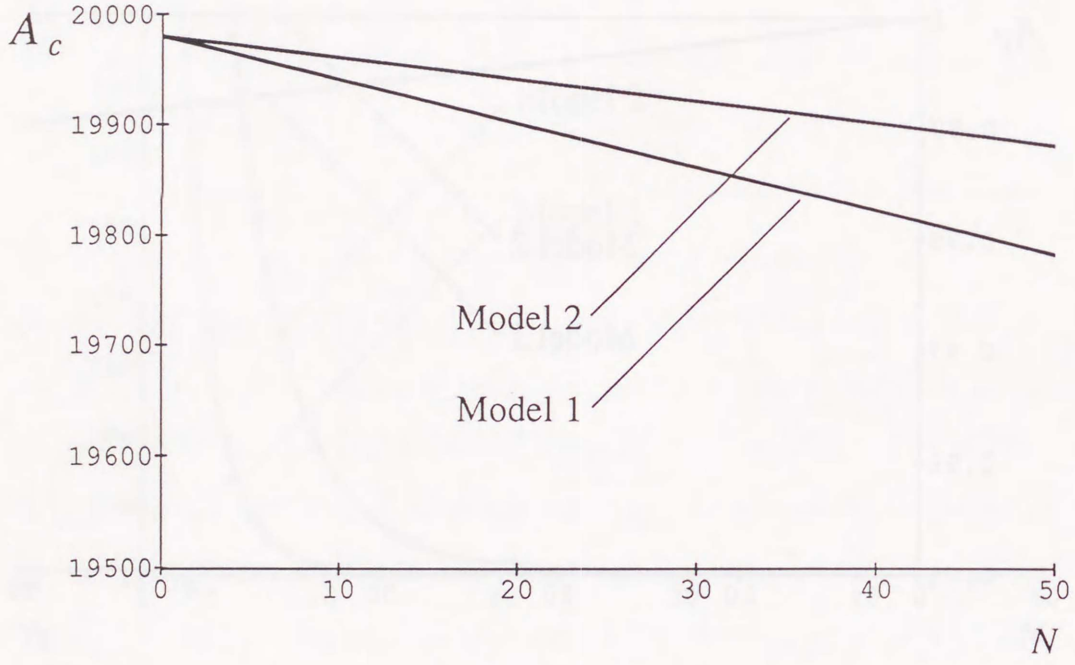

Figure 4.6: The dependence of the capacity of the buffer N on the A_C for $\lambda_2 = 0.0001$.

the parameters are specified just same as in Fig. 4.5. When N is small, C_J of Model 1 is less than that of Model 2. However, when N is large, C_J of Model 1 is greater than that of Model 2. The turning position is $N = 16$. This measure shows the trade-off between reliability and performance in varying N. When N increases, the reliability decreases as shown in Fig. 4.5 and the performance increases as shown in Fig. 4.4. The trade-off yields the critical value of N which minimizes C_J. The critical value of N is 11 and 23 for Models 1 and 2, respectively. It is obvious that the critical value of N of Model 2 is greater than that of Model 1 because of its *fault-tolerance*. From the above results, the random assignment (i.e., Model 2) is superior to the uniform assignment (i.e., Model 1) when the value of N is high.

4.6 Concluding Remarks

In this chapter, we have discussed a multi-processor system composed of two processors and the buffer(s) and proposed two models from the viewpoint of transaction assignments. We further have compared two models using the numerical examples of the reliability/performance measures and shown the characteristics of each model.

In general, the reliability and performance of the computing system is mutually contradictory. For instance, a massive redundant system can attain the high reliability by sacrificing the high performance. From this viewpoint, a multi-processor system should be recommended for both of reliability and

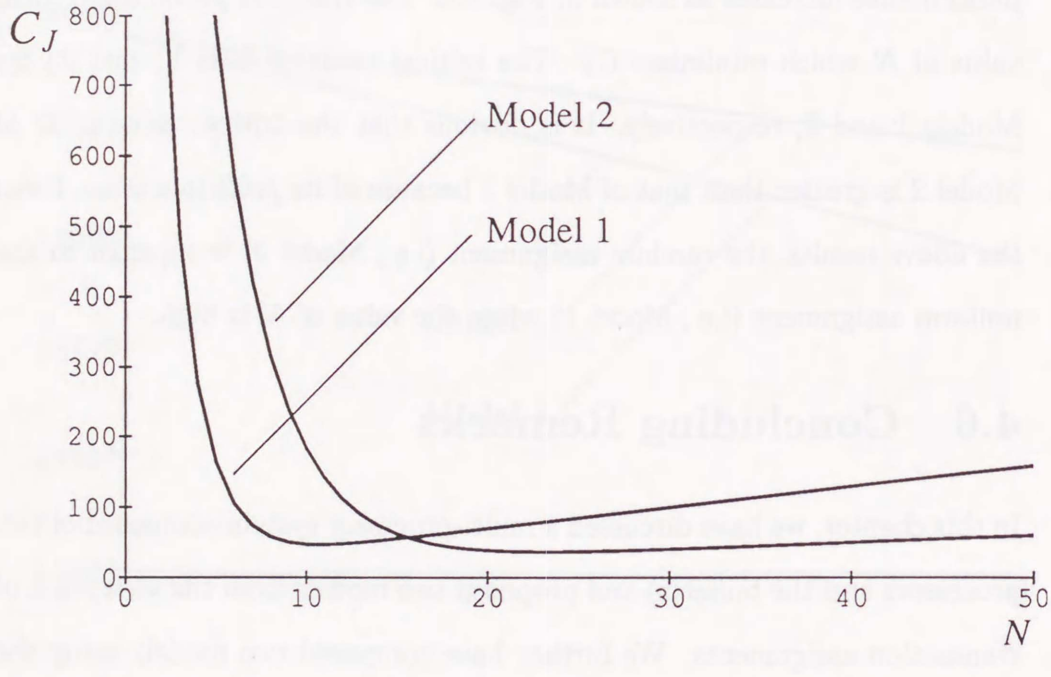

Figure 4.7: The dependence of the capacity of the buffer N on the C_J for $\lambda_2 = 0.0001$.

performance. We have proposed two models from the viewpoint of transaction assignments. We have shown some fruitful conclusions. That is, when the storage capacity of the buffer is small, the uniform assignment of Model 1 is better than the random assignment of Model 2, and, conversely, when the storage capacity is large, the random assignment is better than the uniform assignment. From a practical point of view, this implies that we should compose the two unit multi-processor system by two independent buffers for a large scale application such as on-line transaction processing.

It seems that we can expand our models as follows: Model 1 is composed of m processors and a buffer with maximum capacity mN. Model 2 is composed of m processors and m buffers with maximum capacity N. However, it is too difficult to analyze these extended models by applying Markov renewal processes and queueing theory. Some particular assumptions must be made, e.g., a two-out-of-m: F system.

The assertion of the above conclusions are based on our models and numerical computations. In particular, under the assumption of Model 2, one buffer can overflow although the other buffer is able to accept the arriving transactions. This fact may not fit an actual situation, however it is too difficult to analyze Model 2 taking account of such a situation. Comparisons should be done more precisely by applying the existing and forthcoming sophisticated techniques. Our assertion is a first stage of mathematical modeling for reliability/performance evaluation.

References

[1] A. Avižienis: "Fault-Tolerant System", *IEEE Trans. Comput.*, Vol. C-25, No. 12, pp. 1304-1311 (1976).

[2] D. P. Siewiorek and R. S. Swarz (eds.): *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, Massachusetts (1982).

[3] A. Avižienis and J. C. Laprie: "Dependable computing: From concepts to design diversity", *Proc. IEEE,* Vol. 74, No. 5, pp. 629-638 (1988).

[4] A. Avižienis, H. Kopetz and J. C. Laprie (eds.): *The Evolution of Fault-Tolerant Computing*, Springer-Verlag, Wien (1988).

[5] E. D. S. Silva and H. R. Gail: "Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization", *J. ACM*, Vol. 36, No. 1, pp. 171-193 (1989).

[6] A. Costes, C. Landrault and J. C. Laprie: "Reliability and Availability Models for Maintained Systems Featuring Hardware Failure and Design Faults", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 548-560 (1978).

[7] Y. W. Ng and A. Avižienis: "A Unified Reliability Model for Fault-Tolerant Computers", *IEEE Trans. Comput.*, Vol C-29, No. 11, pp. 1002-1011 (1980).

[8] S. Osaki and T. Nishio: *Reliability Evaluation of Some Fault-Tolerant Computer Architectures*, Springer-Verlag, Berlin (1980).

[9] M. D. Beaudry: "Performance-Related Reliability Measures for Computing Systems", *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 540-547 (1978).

[10] J. F. Meyer: "On Evaluating the Performability of Degradable Computing Systems", *IEEE Trans. Comput.*, Vol. C-29, No. 8, pp. 720-731 (1980).

[11] F. A. Gay and M. L. Ketelsen: "Performance Evaluation for Gracefully Degrading Systems", in *Proc. 9th FTCS*, Madison, Wisconsin, pp. 51-58 (1979).

[12] M. Nakamura and S. Osaki: "Performance/Reliability Evaluation of a Multi-Processor System with Computational Demands," *Int. J. System Sci.*, Vol. 15, No. 1, pp. 95-105 (1984).

[13] T. Nakagawa and S. Osaki: "Stochastic Behavior of a Two-Unit Standby Redundant System", *INFOR*, Vol. 12, No. 1, pp. 66-77 (1974).

[14] T. Nakagawa and S. Osaki: "Markov Renewal Processes with Some Non-Regeneration Points and Their Applications to Reliability Theory", *Microelectron. Reliab.*, Vol. 15, pp. 633-636 (1976).

[15] L. Kleinlock: *Queueing Systems; Volume I: Theory*, John Wiley and Sons, New York (1975).

Chapter 5

A Reliability Evaluation Software Package Tool for Markov Models with Many States

5.1 Introduction

It is of great interest and importance to operate a computing system with high reliability and performance [1, 2]. To evaluate such a system, we should derive analytically and/or numerically reliability/performance measures by formulating a stochastic model of the system [3-5]. A continuous-time Markov chain is one of the most powerful stochastic processes to analyze the system. In particular, we are very much interested in a continuous-time Markov chain with many states since modeling a Markov chain yields many states in practice [6]. We develop a software package tool for calculating the transient state probabilities for a continuous-time Markov chain with many states. Several performance/reliability measures can be calculated by using the state probabilities.

We adopt the randomization technique [7] for calculating the transient state probabilities as well as the steady-state probabilities. It is assumed that the transient state probabilities converge the steady-state probabilities as time tends to infinity under certain assumptions. In principle, it is possible to calculate the transient and steady state transition probabilities. However, it is quite difficult to do so if there are many states such as some hundreds or thousands of states.

For our software package tool, we specify the initial state probability vector $\boldsymbol{\pi}(0)$. Once the initial state probability vector $\boldsymbol{\pi}(0)$ is specified, we can calculate the transient state probability vector $\boldsymbol{\pi}(t)$ at time t. However, it is quite difficult in advance to identify when the transient state probability vector converges to the steady-state probability vector with enough precision. We propose a new idea of calculating the convergence time t_s of the steady-state probability in advance from the knowledge of the randomization technique.

In this chapter, we discuss our software package tool and its applications. In Subsect. 5.2, we discuss the randomization technique for calculating the transient solutions as well as the steady-state solutions for a continuous-time Markov chain with many states. We propose a new idea of the convergence time which will be implemented in our software package tool. We further present two examples of maintenance policies for a computing system in Subsect. 5.3, and show how our software package tool is useful. The first example is maintenance policies based on retries for a computing system, which has been discussed in Chapter 3. Calculating the transient and steady-state availabilities, we can obtain the effective maintenance policies. The second example discusses maintenance policies for a hardware and software system. We propose two software maintenance policies for a two-unit hardware system and compare them.

5.2 Mathematical Preliminaries

5.2.1 Randomization Technique

Let us briefly sketch the randomization technique (see Ross [6], pp 141-183). There are several techniques of calculating the exponential of matrix [8], since they are quite famous as the eigen value problems of the matrices.

Consider a continuous-time Markov chain with N states. Let

$$\pi(t) = \{\pi_1(t), \pi_2(t), \ldots, \pi_N(t)\} \tag{5.1}$$

be the state probability vector at time t, where the initial state vector

$$\pi(0) = \{\pi_1(0), \pi_2(0), \ldots, \pi_N(0)\} \tag{5.2}$$

is prespecified. Let Q be the infinitesimal generator for the continuous-time

Markov chain. Then, the matrix differential equation is given by

$$\frac{d\pi(t)}{dt} = \pi(t)Q, \tag{5.3}$$

where the initial condition $\pi(0)$ is given. Note that each element of the infinitesimal generator is given by

$$q_{ij} = \lim_{\Delta t \to 0} \frac{P\{X(t+\Delta t) = j | X(t) = i\}}{\Delta t}, \quad (i \neq j) \tag{5.4}$$

$$q_{ii} = -\sum_{i \neq j}^{N} q_{ij}. \quad (i = j) \tag{5.5}$$

It is easy to solve the Matrix Differential Equation (5.3). We have

$$\begin{aligned} \pi(t) &= \pi(0)e^{Q_t} \\ &= \pi(0)[I + \sum_{n=1}^{\infty} \frac{(Qt)^n}{n!}], \end{aligned} \tag{5.6}$$

where I is an identity matrix.

Specifying Λ such that $\Lambda = \max_i |q_i|$, we transform the matrix Q into the matrix P:

$$P = Q/\Lambda + I. \tag{5.7}$$

We notice that the matrix P is a transition probability matrix and the properties of all states are preserved under the Transformation (5.7). Introduce the n-step transition probability vector $\phi(n)$ for a discrete-time Markov chain with transition probability matrix P. That is,

$$\phi(0) = \pi(0), \tag{5.8}$$

$$\phi(n+1) = \phi(n)\boldsymbol{P}. \quad (n \geq 0) \tag{5.9}$$

Substituting Equation (5.7) into Equation (5.6), we have

$$\pi(t) = \sum_{n=0}^{\infty}\{\frac{(\Lambda t)^n}{n!}e^{-\Lambda t} \cdot \phi(n)\}. \tag{5.10}$$

The right-hand side of Equation (5.10) is the infinite series of the product of the probability mass function of the Poisson distribution with parameter Λt and the n-step transition probability $\phi(n)$.

In practice, instead of infinite series in Equation (5.10), we adopt the finite series

$$\pi^{\varepsilon}(t) \equiv \sum_{n=0}^{T(\varepsilon,t)}\{\frac{(\Lambda t)^n}{n!}e^{-\Lambda t} \cdot \phi(n)\} \tag{5.11}$$

where

$$T(\varepsilon, t) = \min[k : \sum_{n=0}^{k}\frac{(\Lambda t)^n}{n!}e^{-\Lambda t} > 1 - \varepsilon]. \tag{5.12}$$

and ε is an acceptable error which is enough small. Applying Equation (5.11) with the prespecified acceptable error, we can calculate $\pi^{\varepsilon}(t)$, which is the transition probability vector at time t with enough precision. Figure 5.1 shows an illustration of how to calculate $\pi^{\varepsilon}(t)$.

5.2.2 Steady-State Solution

If the continuous-time Markov chain under consideration is regular, there exists the steady-state probability vector π whose solution is given by

$$\pi Q = 0, \tag{5.13}$$

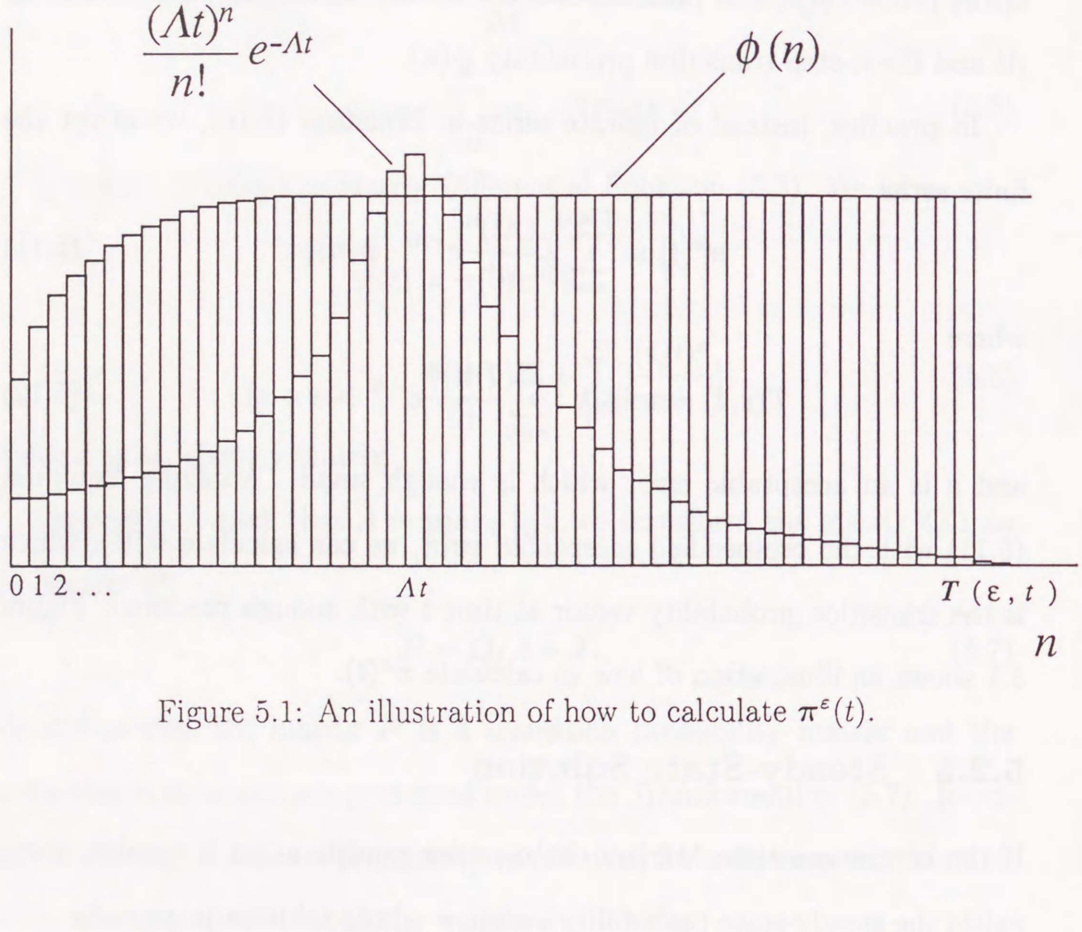

Figure 5.1: An illustration of how to calculate $\pi^{\varepsilon}(t)$.

$$\sum_{i=1}^{N} \pi_i = 1, \tag{5.14}$$

where

$$\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}. \tag{5.15}$$

In principle, it is analytically easy to solve the Linear Simultaneous Equations in (5.13) and (5.14). However, if we consider many states such as several hundreds or thousands of states, we have to consider the efficient method of solving equations (5.13) and (5.14) numerically.

As shown in Equation (5.10), we have to calculate $\phi(n)$. It is easier to obtain the steady-state vector $\phi = \lim_{n\to\infty} \phi(n)$ since the properties of all states are preserved both for the discrete-time Markov chain and the continuous-time Markov chain. That is

$$\lim_{t\to\infty} \pi(t) = \lim_{n\to\infty} \phi(n). \tag{5.16}$$

It is evident that $\phi = \phi P$. Multiplying ϕ for both sides of Equation (5.7), we have

$$\phi Q = 0, \tag{5.17}$$

and $\sum_i \phi_i = 1$, which is a unique steady-state vector π. Let n_s be the minimum step number in which $\phi(n_s)$ approximates ϕ with enough precision. That is,

$$\phi(n_s) = \pi(0) P^{n_s} \simeq \phi = \pi, \tag{5.18}$$

where n_s is the minimum number of the step n such that

$$[\max_j |\phi_j(n) - \phi_j(n-1)|] < \varepsilon_2, \tag{5.19}$$

$$\phi = \{\phi_1, \phi_2, \cdots, \phi_N\}. \tag{5.20}$$

Note that ε_2 is an acceptable error which is prespecified and enough small. From these facts, we can calculate $\phi(n_s)$ instead of π. For the randomization technique, we have to calculate $\phi(n)$ with enough steps which is approximately regarded as the steady-state. We should apply $\phi(n_s)$ in practice.

5.2.3 Convergence Time of Steady-State Solutions

The randomization technique is available for calculating the transient solutions for continuous-time Markov chain. However, it involves an important and difficult problem of identifying when the transient solution converges to the steady-state solution with enough precision. Otherwise, we have to calculate the transient solution by applying the randomization technique in Equation (5.10) which is enormous calculations as time tends to infinity.

Let t_s denote the convergence time such that

$$\pi(t_s) \simeq \pi. \tag{5.21}$$

Then we should calculate $\pi(t)$ for $0 \leq t \leq t_s$. If we have to calculate $\pi(t)$ for $t > t_s$, we should use π instead of $\pi(t)$ in order to avoid the unnecessary calculation.

From Equation (5.18), we assume

$$\phi(n) = \pi \tag{5.22}$$

where $n \geq n_s$. If t_s satisfies $\pi(t_s) \simeq \pi$, almost all the probability mass functions of the Poisson distribution with parameter Λt_s is distributed on n such that $n \geq n_s$. That is,

$$\sum_{n=n_s}^{\infty} \frac{(\Lambda t_s)^n}{n!} e^{-\Lambda t_s} \simeq 1. \tag{5.23}$$

Hence we have the following approximate equation

$$\begin{aligned}
\pi(t_s) &\simeq \sum_{n=n_s}^{\infty} \frac{(\Lambda t_s)^n}{n!} e^{-\Lambda t_s} \cdot \phi(n), \\
&= \sum_{n=n_s}^{\infty} \frac{(\Lambda t_s)^n}{n!} e^{-\Lambda t_s} \cdot \pi, \\
&= \pi \sum_{n=n_s}^{\infty} \frac{(\Lambda t_s)^n}{n!} e^{-\Lambda t_s}, \\
&\simeq \pi.
\end{aligned} \tag{5.24}$$

Let t_s satisfy

$$\Lambda t_s = n_s + k\sqrt{\Lambda t_s}, \tag{5.25}$$

(see Fig. 5.2), where Λt_s and $\sqrt{\Lambda t_s}$ are the mean and standard deviation of the Poisson distribution with parameter Λt_s, and k is a positive real constant. Solving t_s in Equation (5.25), we have

$$t_s = \frac{2n_s + k^2 + k\sqrt{k^2 + 4n_s}}{2\Lambda}, \tag{5.26}$$

That is, t_s is expressed in terms of n_s. In other words, once n_s and k are specified, we can obtain t_s in Equation (5.26). Let us consider how to specify a constant k in Equation (5.25) or (5.26). It is well-known that the Poisson distribution is approximated by the normal distribution when the parameter Λt_s is enough large. Noting this fact, we have

$$\int_{-4}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = 0.9999864 \, . \tag{5.27}$$

In practice, if we assume that $k = 4$, Equation (5.23) is approximated by

$$\sum_{n=n_s}^{\infty} \frac{(\Lambda t_s)^n}{n!} e^{-\Lambda t_s} = 0.9999864 \, . \tag{5.28}$$

which is a good approximation from the viewpoint of round error.

We summarize the convergence time t_s. Once n_s is specified by calculating $\phi(n_s)$ which is approximately the steady-state probability vector, we can calculate t_s from Equation (5.26). That is, if $t > t_s$, we should use $\pi = \phi$ instead of calculating $\pi(t)$ for each t. We emphasize that t_s can be calculated in advance when we implement the randomization technique in our software package tool.

5.3 Applications

5.3.1 Maintenance Policies for a Computing System with Retries

In this subsection, we discuss maintenance policies for a computing system with retries, by applying our software package tool. This model has already

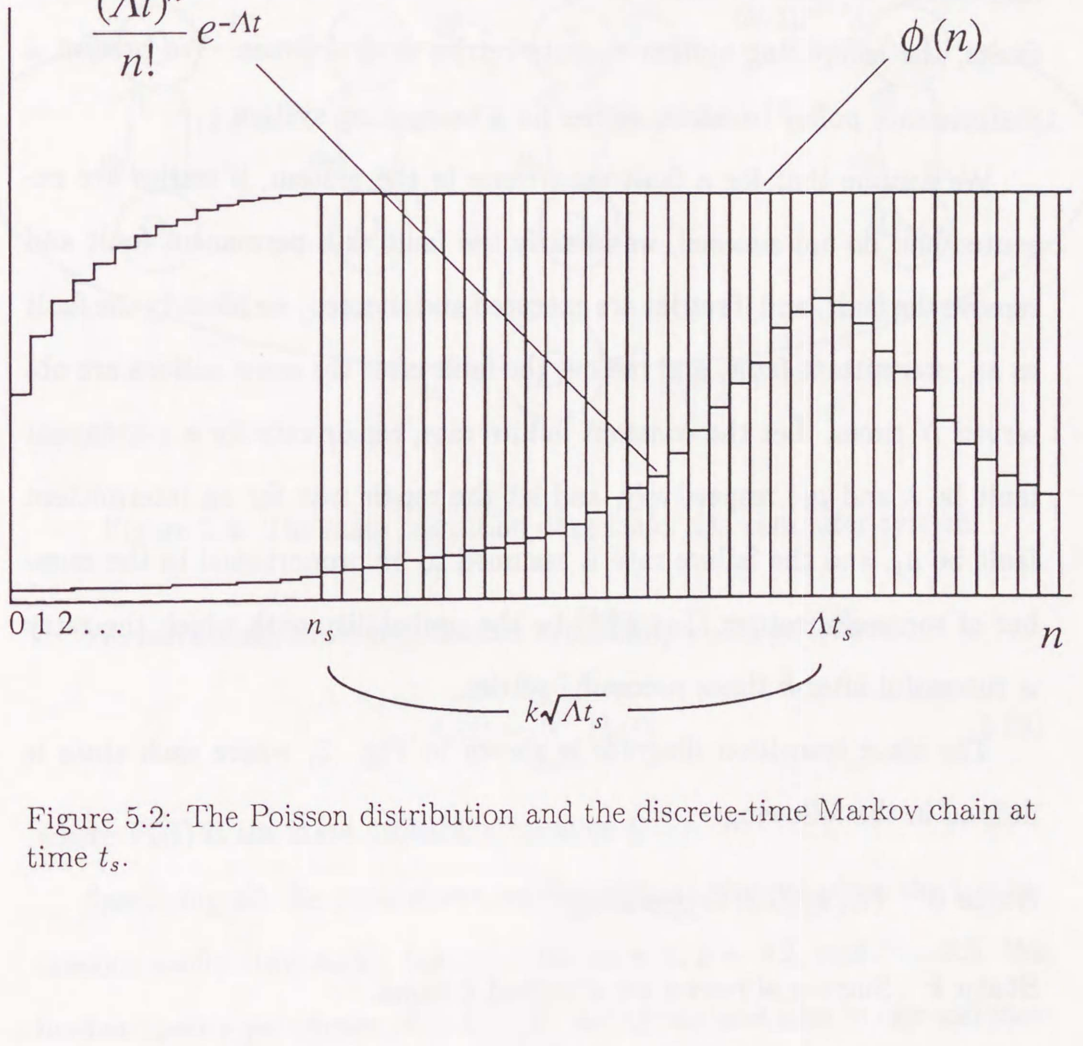

Figure 5.2: The Poisson distribution and the discrete-time Markov chain at time t_s.

been discussed in Chapter 3, and is further simplified here.

It is generally considered that a computing system has two kinds of faults from the viewpoint of maintenance [9]. One is an *intermittent* fault that is only occasionally present. The other is a *permanent* fault that is continuous and stable. In order to avoid the system downs caused by the intermittent faults, the computing system execute retries several times. We present a maintenance policy based on retries for a computing system.

We assume that for a fault occurrence in the system, if retries are executed and do not succeed, we identify the fault as a permanent fault and remove the fault, and if retries are executed and succeed, we identify the fault as an intermittent fault, and remove the fault after the same actions are observed N times. Let the constant failure rate, repair rate for a permanent fault be λ and μ_0, respectively, and let the repair rate for an intermittent fault be μ_1, and the failure rate is assumed to be proportional to the number of successful retries. Let P^{k+1} be the probability with which the retry is successful after k times successful retries.

The state transition diagram is shown in Fig. 3, where each state is defined in the following:

State 0 : The system is operating.

State k : Success of retries are observed k times.

State D_k : Repair for a permanent fault starts (system down).

State D : Success of retries are observed N times and the repair for an intermittent fault starts (system down).

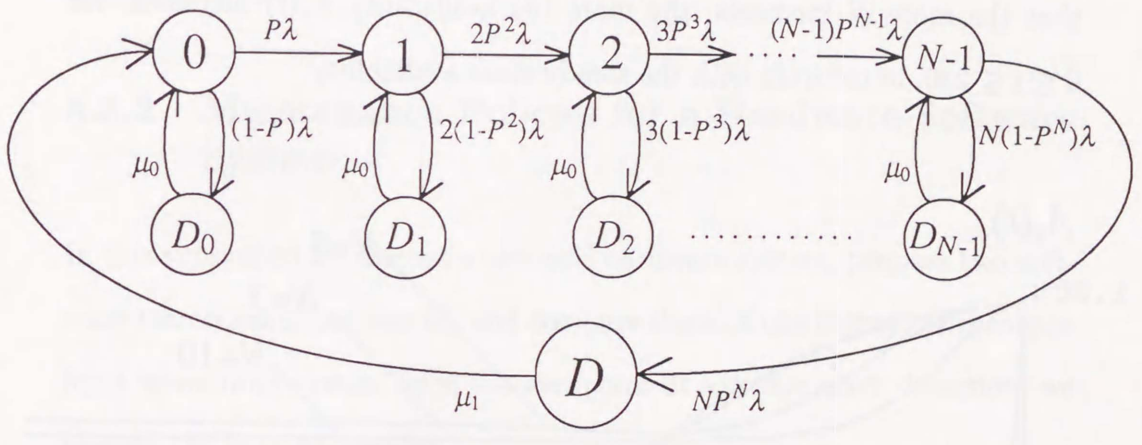

Figure 5.3: The state transition diagram of the computer system.

We can obtain the instantaneous availability $A_v(t)$ as follows:

$$A_v(t) = \sum_{k=0}^{N-1} P_k(t), \tag{5.29}$$

where $P_k(t)$ is the state probability at time t.

Specifying all the parameters, we can numerically calculate the instantaneous availability $A_v(t)$. Let $\lambda = 0.01$, $\mu_0 = 1$, $\mu = 0.2$, and $P = 0.9$. We further specify parameter $N = 1 \sim 20$. Using our new idea in our software package tool, it is shown that the steady-state availability A_v attains the maximum 0.976 at $N = 5$ among all possible N. The convergence time t_s at $N = 5$ is 596. Therefore, we should calculate the transient state availabilities for $0 \leq t \leq 600$.

Figure 5.4 shows the transient state availability $A_v(t)$. It is interesting that the more N increases, the more the availability $A_v(t)$ increases, for $0 \leq t \leq 230$, in contrast with the steady-state availability.

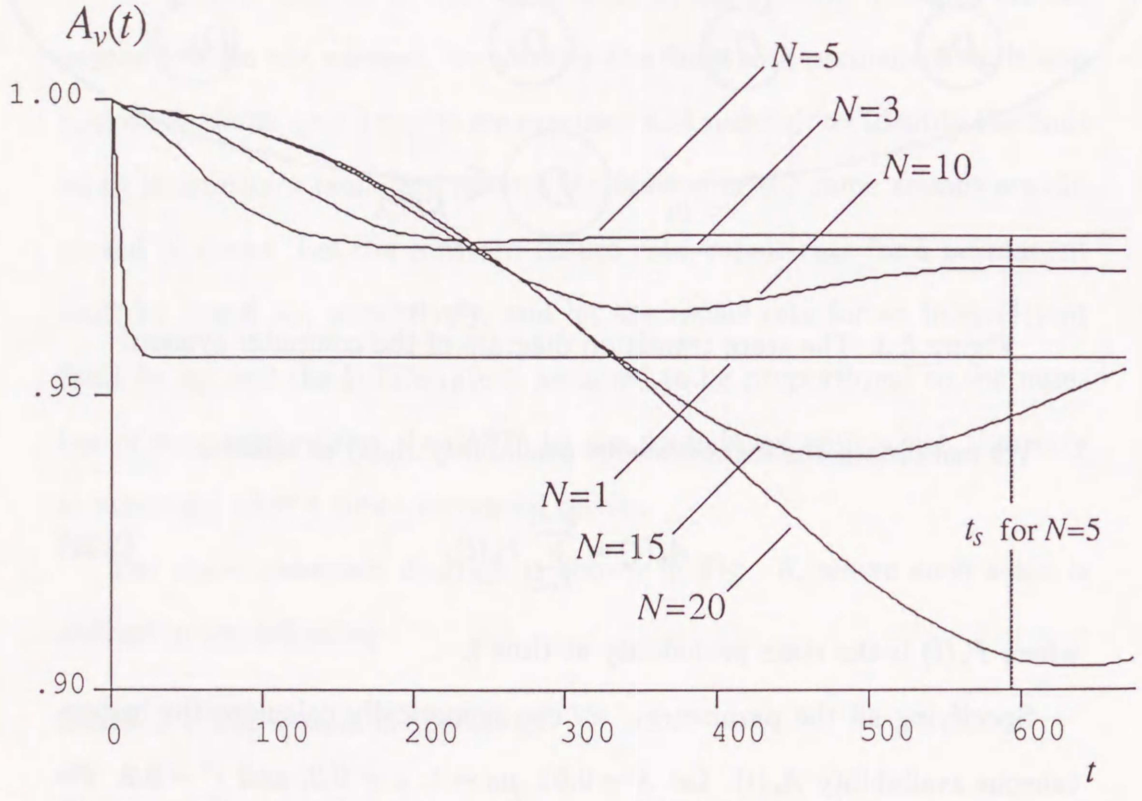

Figure 5.4: The behavior of $A_v(t)$ for each N.

Referring to the above results, we can conclude that we should repair and remove the intermittent fault after successes of retries are observed 5 times, in operation of the system for a long term. However, in operation of the system for a short term like for $0 \leq t \leq 200$, we should not repair the intermittent fault.

5.3.2 Maintenance Policies for a Hardware-Software System

In this subsection we discuss a two-unit hardware system, propose two software maintenance policies [4], and compare them. Considering maintenance for a computing system from the viewpoint of software error detection, we present the following model.

A hardware system is composed of two units. The system can function if only one of the two units functions. Each unit is assumed to be repairable, and the constant failure rates and maintenance rates for each unit are assumed to be λ_0 and μ_0, respectively.

We assume that any software error causes the system down, and that there are N software errors latent at the installation of the software system. The detection time of each software error is assumed to be exponentially distributed and its detection rate is also constant and proportional to the remaining number of errors (see Jelinski and Moranda [10]). Let λ_s and μ_s be the detection and maintenance rate for each software error. It is assumed that there is a single repair or maintenance facility. Here a repair or maintenance is assumed perfect. All the states of the system are defined as follows.

State 0_n : A system starts operating.

State 1_n : A hardware failure of a unit takes place and its repair starts.

State 2_n : Through state 1_n, a hardware failure of the remaining unit takes place (system down).

State 3_n : Through state 0_n, a software error takes place (system down).

State 4_n : Through state 1_n, a software error takes place (system down).

Here n denotes the number of remaining error at that time, where $n = 0, 1, \cdots, N$.

We are now ready to introduce the following software maintenance policies through state 4_n.

Model 1 : After the repair completion of hardware, the software maintenance starts.

Model 2 : The software maintenance starts even if the repair of hardware is interrupted; the interrupted hardware repair restarts after the software error maintenance completion.

From the above definitions, for Model 1 the process can move to state 3_n from state 4_n, and for Model 2 the process can move to state 1_{n-1} from state 4_n. The state transition diagram is shown in Fig. 5.5.

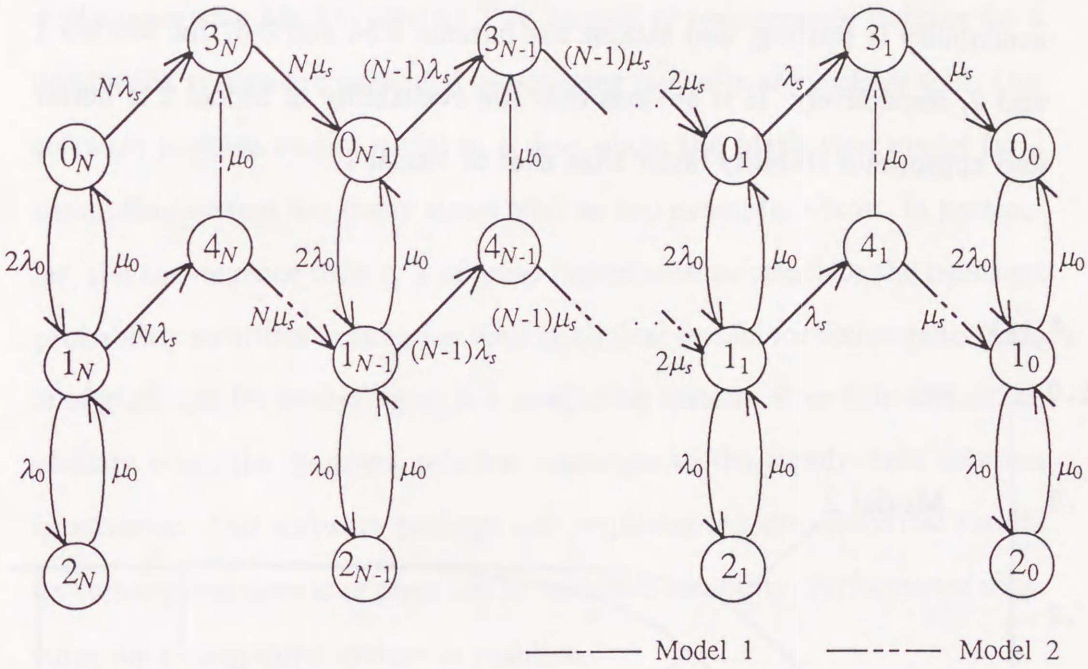

Figure 5.5: The state transition diagram for each model.

We can obtain the instantaneous availability $A_v(t)$ as follows:

$$A_v(t) = \sum_{k=0}^{N} [P_{0_k}(t) + P_{1_k}(t)] \tag{5.30}$$

where $P_{0_k}(t)$ or $P_{1_k}(t)$ is the state probability at time t.

Specifying all the parameters, we can numerically calculate the instantaneous availability $A_v(t)$. Let $N = 10$, $\lambda_s = 0.02$, $\mu_s = 0.05$, $\lambda_0 = 0.05$, and $\mu_0 = 0.025$. Since t_s for Model 1 is 1003 and t_s for Model 2 is 871 by our software package tool, we should calculate the transient probabilities for $0 \leq t \leq 1000$.

Figure 5.6 shows the availability $A_v(t)$. Just after the installation, the availability is settling, and attains the minima 0.54 and 0.67 for Models 1 and 2, respectively. It is obvious that the availability of Model 2 is better and approaches stability faster than that of Model 1.

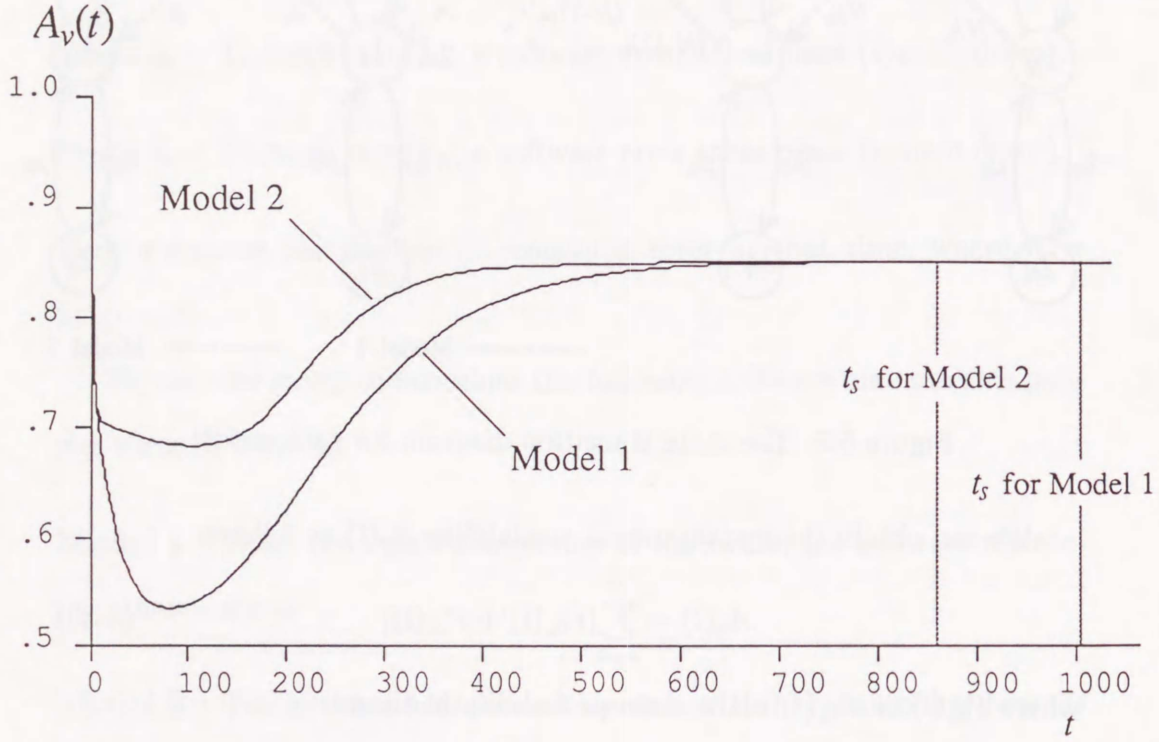

Figure 5.6: The behavior of $A_v(t)$ for each model.

5.4 Concluding Remarks

In this chapter, we have discussed the randomization technique for calculating the transient solutions as well as the steady-state probability for a continuous-time Markov chain. Two models of maintenance policies for a computing system are presented by applying our software package tool. Our software package tool is useful in a case where the Markovian model for a computing system has many states such as two examples above. In particular, the convergence time t_s is of great importance to calculate the transient probability solutions in practice. Our analytical results for convergence time is of great use for analyzing such a computing system, since it is difficult to identify when the transient solution converges to the steady-state solution in advance. Our software package tool implemented the analytical results on convergence time is of great use to calculate reliability/performance measures for a computing system in practice.

References

[1] A. Avizienis: "Fault-Tolerant System", *IEEE Trans. Comput.*, Vol. C-25, No. 12, pp. 1304-1312 (1976).

[2] D. P. Siewiorek and R. S. Swarz (eds.): *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, Massachusetts (1982).

[3] S. V. Makam and A. Avizienis: "ARIES: A Reliability and Life-Cycle Evaluation Tool for Foult-Tolerant System", in *Proc. 12th FTCS*, Santa Monica, California, pp. 267-274 (1982).

[4] A. Avizienis and J. C. Laprie: "Dependable Computing: From Concepts to Design Diversity", *Proc. IEEE*, Vol. 74, No. 5, pp. 629-638 (1986).

[5] J. Arlat, K. Kanoun and J. C. Laprie: "Dependability Evaluation of Software Fault-Tolerance", in *Proc. 18th FTCS*, Tokyo, pp. 142-147 (1988).

[6] S. M. Ross: *Stochastic Processes*, John Wiley and Sons, New York (1983).

[7] D. Gross and D. R. Miller: "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes", *Oper. Res.*, Vol. 32, No. 2, pp. 343-361 (1984).

[8] C. Moler and C. Van Loan: "Nineteen Dubious Ways to Compute the Exponential of Matrix", *SIAM Rev.*, Vol. 20, No. 10, pp. 801-836 (1978).

[9] S. Y. H. Su, I. Koren and Y. K. Malaiya: "A Continuous-Parameter Markov Model and Detection Procedures for Intermittent Faults", *IEEE Trans. Comput.*, Vol. C-27, No. 1, pp. 567-570 (1978).

[10] Z. Jelinski and P. B. Moranda: "Software Reliabilty Research", in *Statistical Computer Performance Evaluation*, W. Freiberger (ed.), Accademic Press, New York, pp. 465-484 (1972).

Chapter 6

Conclusion

This dissertation has been discussed configuration and recovery techniques for fault-tolerant computing systems. We have presented stochastic models for the techniques to evaluate performance and/or reliability, and discussed a numerical calculation method for reliability evaluation.

In Chapter 2, database recovery has been discussed. First, a model evaluating the recovery mechanism has been presented in order to treat the changing of a failure rate of the system with time. In this model, one cycle has been described as the interval from the start of the system operation to the completion of recovery action after the failure. Impacts of checkpoint intervals on the availability for one cycle has been estimated analytically or numerically. Secondly, a model evaluating the recovery mechanism has been presented in the situation where the road of the system varies with time in a shape of a cycle. The density of checkpoint generations, measured in unit of update pages, has been derived so as to minimize the expected total overhead to completion of a phase.

In Chapter 3, we have been discussed a model for evaluating the improvement on reliability by retries. This model has assumed to manage the system maintenance with prescribed number of successful retries, taking account of the behavior of intermittent and permanent failures. The availability and the mean time between failures in the steady-state, and the availability in the transient-state have been evaluated analytically and numerically.

In Chapter 4, we have discussed multi-processor systems which have been assumed to be composed of two processors and buffer(s). Two models have been proposed from the viewpoint of transaction assignment, and have been compared using the reliability/performance measures.

Finally, Chapter 5 has discussed a reliability evaluation software package tool for a system formulated by a continuous-time Markov chain with many states. The randomization technique, which calculates a transient solution for the Markov chain, has been used to develop the software package tool. We have further introduced a new idea of identifying when the transient solution converges to the steady-state solution in advance. Two examples of maintenance models for a computer system have been shown by our software package tool.

The main contributions obtained in the dissertation are shown as follows:

(1) A new formula of the system availability for evaluating a database recovery mechanism is derived in the situation where the failure rate of

the system changes with time. Moreover, assuming a constant failure rate, a formula of the optimum checkpoint interval for a database recovery mechanism is derived as a generalized version of the one for a endless job processing. Numerical examples of these formulae explain that the mean time to the failure is the dominating cause of the optimum checkpoint interval more than the shape of the failure rate variation.

(2) In the situation where load of the database system changes with time, a new model is proposed for determining the sequence of checkpoint generations measured in unit of update pages so as to minimize the expected total overhead. While the previous works discussing such situations have presented the algorithms with enormous calculation, an analytically efficient result obtained here yields the optimum sequence relatively easily.

(3) A maintenance policy for a computer system is presented, which removes intermittent failures with prescribed number of successful retries. This model enables us to evaluate the reliability in the transient-state as well as the steady-state.

(4) The impacts of transaction assignment on reliability/performance evaluation of multi-processor systems are examined. The analytical and numerical results imply that when the storage capacity of the buffer

is small, the uniform assignment is better than the random assignment, and, conversely, when the storage capacity is large, the random assignment is better than the uniform assignment.

(5) A new idea is presented to identify when the transient solution of a continuous-time Markov chain converges to the steady-state solution in advance. Once the initial state probability vector and the infinitesimal generator of the Markov chain are specified, we can calculate the convergence time t_s, and can restrict the calculation for the transient solution $\pi(t)$ to the range of $0 \leq t \leq t_s$.

Most results derived in this dissertation are analytical results. Thus, we can numerically evaluate the models relatively easily, specifying each parameter of the results. In the numerical illustrations of this dissertation, we have discussed the mean time to the failure, the mean overhead time for the maintenance, and so on, based on the results in the References [3, 10] of Chapter 1.

Generally, in a mathematical modeling, including a stochastic modeling, various restrictions are imposed on composing a model, comparing with a simulation modeling. However, once the modeling can be implemented, a mathematical modeling rather becomes an advantage of a computation treatment, a sensitivity test, and so on.

We finally shows the future research works concerned with the configuration and recovery techniques for fault-tolerant computing systems as follows:

- Fuzzy checkpoints or other advanced checkpoints for a database recovery mechanism (Reference [2] of Chapter 2) should be also discussed, which we have not treated. Adoption of these checkpoints will gradually increase in the actual systems, since they have the advantage of low overhead during normal operations.

- An evaluation model for a recovery mechanism in a distributed system, in which their databases are brought to consistent states after the failure (Reference [25] of Chapter 1), should be examined. Our models in this dissertation have focused the attention on an independent database.

- The failure modes of the intermittent failures should be further studied. In particular, we have not obtained the reasonable description for the situation where an intermittent failure turns into a permanent failure.

- Two models for two unit multi-processor systems presented in this dissertation may be the first stages of modeling for generalized multi-processor systems, that is, *a tightly coupled multi-processor system* and *a loosely coupled multi-processor system.* These systems could be examined by some evaluation technique for reliability/performance.

- Some approach for a total system modeling is expected to develop, which can take account of some or all configuration and recovery techniques, fault detection techniques, diagnosis techniques, simultaneously.

These works seem to involve many problems to be solved. In particular, the evaluation for a recovery mechanism in a distributed database system is presumed to yield very complicated models, since the system has a network which causes the delay and/or interrupt of the data communication. The first approach for the above works will be a simplified description of each system component, and selections of the dominating components of the evaluation measures in construction of the models.

PUBLICATIONS LIST OF THE AUTHOR

[1] S. Osaki and S. Fukumoto: "Derivation of the Convergence Time of the Limiting Probabilities for a Markov Process", *Trans. IEICE of Japan*, Vol. J71-A, No. 4, pp. 1062-1065 (1988) (in Japanese).

[2] S. Fukumoto and S. Osaki: "Numerical Calculation of Transient Solutions for a Markov Process with Many States", in *Abstract of The First Conference of the Association of Asian-Pacific Operational Research Societies*, p. 27, Seoul (1988).

[3] S. Osaki, H. Ohshimo and S. Fukumoto: "Effect of Software Maintenance Policies for a Hardware-Software System", *Int. J. System Sci.*, Vol. 20, No. 2, pp. 331-338 (1989).

[4] S. Fukumoto and S. Osaki: "Maintenance Policies Based on Retries for a Computer System", *Trans. IEICE of Japan*, Vol. J73-D-I, No. 2, pp. 161 -169 (1990) (in Japanese).

[5] Satoshi Fukumoto and Shunji Osaki: "A Software Package Tool for Markovian Computing Models and Its Applications", in *Proc. of the IEEE International Phoenix Conference on Computers and Communications*, pp. 872-873, Phoenix, Arizona (1990).

[6] S. Fukumoto, N. Kaio and S. Osaki: "Optimal Checkpointing Policies

Using the Checkpointing Density", *Trans. IPS Japan*, Vol. 31, No. 6, pp. 887-893 (1990) (in Japanese).

[7] H. Ohshimo, S. Fukumoto and S. Osaki: "Reliability/Performance Evaluation for Multisystems from the Viewpoint of Job Assignments", *Trans. IEICE of Japan*, Vol. E-73, No. 8, pp. 1257-1263 (1990).

[8] S. Fukumoto, N. Kaio and S. Osaki: "Evaluation for a Database Recovery Action with Periodical Checkpoint Generations", *IEICE Trans. of Japan*, Vol. E-74, No. 7, pp. 2076-2082 (1991).

[9] S. Fukumoto and S. Osaki: "A Software Package Tool for Markovian Computing Models with Many States: Principles and Its Applications", *Stochastic Processes and Their Applications*, edited by M. J. Beckmann, M. N. Gopalan and R. Subramanian, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Heidelberg, pp. 222-231 (1991).

[10] S. Fukumoto, N. Kaio and S. Osaki: "A Study of Checkpoint Generations for a Database Recovery Mechanism", to appear in *Computers & Mathematics with Applications* (1992).