General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

CLOSED-FORM SOLUTIONS OF PERFORMABILITY *

J. F. Meyer, Senior Member, IEEE

Depatment of Electrical and Computer Engineering
Department of Computer and Communication Sciences
The University of Michigan
Ann Arbor, Michigan 48109

Abstract

If computing system performance is degradable then, as recognized in a number of recent studies, system evaluation must deal simultaneously with aspects of both performance and reliability. One approach is the evaluation of a system's "performability" which, relative to a specified performance variable Y, generally requires solution of the probability distribution function of Y. In this paper we examine the feasibility of closed-form solutions of performability when Y is continuous. In particular, we consider the modeling of a degradable buffer/multiprocessor system whose performance Y is the (normalized) average throughput rate realized during a bounded interval of time. Employing an approximate decomposition of the model, we show that a closed-form solution can indeed be obtained.

I. INTRODUCTION

In the evaluation of computing systems, issues of performance and reliability have traditionally been distinguished by regarding "performance" as "how well the system performs, provided it is correct" (see [1]-[3], for example) and regarding "reliability" as "the probability of performing successfully" (see [4]-[7], for example). Although this distinction is meaningful for hardware and software architectures which exhibit "all or nothing" behavior in the presence of faults, it becomes blurred in the context of distributed, multifunction systems (computers, computer-communication networks, operating systems, data bases, etc.) where performance is "degradable." As recognized in a number of recent studies [8]-[16], the evaluation of degradable systems calls for unified performance-reliability measures which, in the terminology of [12], quantify a system's "performability." Such measures, in turn, call for appropriate generalizations of the types of analytic models and solution methods employed in performance and reliability evaluation.

* This work was supported by the NASA Langley Research Center under Grant No. NSG 1306.

To accommodate these needs, a general modeling framework was introduced in [8] (and subsequently refined in [12]) wherein the "performance" of a system S over a specified time period T is represented by a random variable Y_S taking values in a set A. Elements of A are the "accomplishment levels" (performance outcomes) to be distinguished in the evaluation process. With respect to Y_S, the "performability" of S is the probability measure induced by Y_S where, for any measurable set B of accomplishment levels ($B \subseteq A$),

$p_S(B)$ = probability that S performs at a level in B.

(For more precise development of this and related concepts, see [12].) Performability evaluation thus entails a complete probabilistic description of the performance variable Y_S, as opposed to partial information such as its expected value, its variance, etc.

Prior work on the development of specific performability models and solutions has dealt primarily with discrete performance variables ranging over a countable and typically finite set of accomplishment levels (see [11], [14], for example). In the overall process of system design and validation, the use of these discrete variable methods is best suited to validation of a completed system design with respect to "bottom line" performability requirements. However if the evaluation results disclose that a design is deficient, the performability data need not be indicative of just how the design should be modified. This is due to the fact that lower level, design-oriented details are often suppressed by a high level, discrete performance variable. Hence early validation (during the design process) at lower system and subsystem levels is required if negative results are to indicate how the design should be modified.

In the latter validation context and, more generally, in the context of "design aids," we believe that performability models and solutions can likewise play an important role. Here, there is a need to consider more detailed aspects of system and subsystem behavior(e.g., speed, responsiveness, etc.) which, when modeled as performance variables, can assume a continuum of values. Accordingly, the evaluation methods called for here must deal with continuous performance variables as well as discrete performance variables. Moreover, to support the investigation of various design trade-offs, there is a need to develop methods which yield closed-form performability solutions, expressed as a function of the underlying model parameters. Some initial results, aimed at fulfilling

these needs, are established in the presentation that follows. The presentation expands on an earlier, more condensed description of this work that appeared in [17].

II. MODEL CONSTRUCTION

The system we consider is a total system S = (C,E) where, informally, computer C and environment E can be described as follows. C is a degradable multiprocessor system consisting of N identical processors ($N \geq 1$) and a buffer (queue) for temporary storage of incoming tasks (see Fig. 1). The buffer B is assumed to have a finite capacity L ($L \geq 0$), that is, B is capable of storing at most L tasks. Note that, by allowing N = 1, we are including the degenerate case of a (nondegradable) uniprocessor system. Similarly, by allowing L = 0, we are including the case where C actually has no buffer at all. The environment E is the arrival of computational tasks at the input to the computer. We assume here that tasks arrive randomly (one at a time) and that there is no upper bound on the total number of arrivals. More detailed descriptions of C and E will be supplied momentarily.

Performance Variable

Regarding performance, we presume that, ideally, the user wants the computer to process all tasks that arrive during some specified utilization period T. However, due to the finite capacity of the buffer and to faults which may occur in the buffer and processors, ideal behavior will generally not be attainable. Accordingly, an interesting measure of performance in this context is the fraction of task arrivals that C in fact processes during utilization. To define this more precisely, if $t \in [0,\infty)$, let A_t = number of tasks that arrive during [0,t] and let D_t = number of tasks that are processed during [0,t]. Then, relative to the utilization period T = [0,t], we take the performance of S to be the random variable

$$Y_S = \frac{D_t}{A_t} = \text{fraction of arrived tasks processed during T.} \qquad (1)$$

Alternatively, if we let

$$\alpha_t = \frac{A_t}{t} = \text{average arrival rate during [0,t]} \qquad (2)$$

$$\delta_t = \frac{D_t}{t} = \text{average throughput rate of S during } [0,t] \tag{3}$$

then

$$Y_S = \frac{D_t}{A_t} = \frac{D_t/t}{A_t/t} = \frac{\delta_t}{\alpha_t} = \frac{\text{average throughput rate of S during T}}{\text{average arrival rate during T}} \tag{4}$$

In other words, the performance of S can also be interpreted as the "normalized average throughput rate," normalized with respect to the average arrival rate and averaged over the utilization period T = [0,t].

To solve the probability distribution function of Y_S and, hence, the performability of S, the specific nature of the computer C and environment E must be spelled out in more detail. We begin with the environment.

Environment Model

If, as above, we let A_t denote the number of task arrivals during the interval [0,t], the environment E can be regarded as a stochastic process $X_E = \{A_t | t \in [0,\infty)\}$ where the variables A_t take values in the state set $Q_E = \{0,1,2,\ldots\}$. To designate the specific nature of X_E, we suppose further that arrivals are "purely random" in the sense that interarrival times are independent random variables with identical exponential distributions. This is equivalent to saying that the arrival process X_E is a Poisson process. Accordingly if we let

$$\alpha = \text{average arrival rate (in the long run)}, \tag{5}$$

that is, $\alpha = \lim_{t\to\infty} \alpha_t$ (see (2)), then X_E is uniquely determined by α.

Computer Model

As depicted in Fig. 1, the fault-free structure of the computer is determined by values of two basic parameters:

$$N = \text{number of processors } (N \geq 1) \tag{6}$$

$$L = \text{storage capacity of the buffer } (L \geq 0). \tag{7}$$

To describe how the system is altered by faults, we assume the following. If C is fault-free (i.e., resources B, P_1, P_2,..., P_N are fault-free) then all processors are active (no "stand-bys") and are able to process tasks concurrently. Each pro-

cessor is self-testing and, in the presence of a single faulty processor, the system is able to recover (with a specified "coverage") to an (N-1)-processor configuration, provided $N \geq 2$. In this configuration, C behaves the same as a fault-free version of the system with N-1 processors (provided $(N-1) \geq 2$). When only a single processor remains fault-free, fault recovery is no longer possible. The input buffer B is assumed to be nondegradable, i.e., it either performs correctly or fails. (In a more general example, the buffer could likewise be treated as a degradable resource.) Either failure to recover from a processor fault or failure of the buffer results in a total loss of processing capability (system failure).

Under the above assumptions, the relevant structural configurations of C can be represented by the state set $Q_R = \{0,1,...,N\}$ where state i is interpreted as follows:

i : fault-free buffer and i fault-free processors ($1 \leq i \leq N$)

0 : system failure

Modeling how structure varies (probabilistically) as a function of time thus reduces to a standard problem encountered in reliability modeling. In this regard, let us assume that resources fail (become faulty) at constant rates equal to their respective long run average failure rates. More specifically, for each of the processors, let

λ_P = processor failure rate (8)

and let c_P denote the coverage referred to above, i.e.,

c_P = probability of recovering from a processor fault. (9)

For the buffer B with capacity L, we assume that B is constructed from L "stages" (a stage can store a single task) where, for each stage,

λ_b = buffer stage failure rate. (10)

Then, if stages fail independently and any stage failure results in a buffer failure, it follows that

λ_B = buffer failure rate = $L\lambda_b$. (11)

Finally, if we suppose that resource failures are independent and permanent

(i.e., there is no "repair") then the structure of C can be modeled as the Markov process X_R of Fig. 2, where the state set of X_R is Q_R.

The parameters $\lambda_i (1 \leq i \leq N)$ and $c_i (2 \leq i \leq N)$ of Fig. 2 are formulated as follows in terms of the basic parameters defined above. The transition rate λ_i from structure state i is just the accumulated failure rate of fault-free resources associated with state i, that is,

$$\lambda_i = i\lambda_P + \lambda_B = i\lambda_P + L\lambda_b \,. \tag{12}$$

The combined "coverage" c_i in state i (when interpreted directly in terms of Fig. 2) is the probability of a transition to state i-1 given a transition from state i. In terms of resource faults, c_i is therefore the probability that a transition from state i is caused by a processor fault and, in turn, C is able to recover from that fault via self-test and reconfiguration. As the latter is specified by the coverage parameter c_P (9), it follows, via a simple conditional probability argument, that

$$c_i = \frac{i\lambda_P c_P}{i\lambda_P + L\lambda_b} = \frac{c_P}{1 + \dfrac{L\lambda_b}{i\,\lambda_P}} \,. \tag{13}$$

For each structure state $i \in Q_R$, we now proceed to construct a submodel of C that accounts for the internal state behavior of C when its structure is fixed at i. Suppose, first, that all resources are fault-free (i=N). Then when the buffer is empty and at least one processor is idle, processing of an incoming task is immediately undertaken by an idle processor. If all processors are busy, an incoming task is stored in buffer B, provided B is not "full" (i.e., the number of tasks stored in B is less than L); as soon as one of the processors becomes idle, it begins to process the task that was least recently stored in the buffer. Finally, if B is full when a task arrives, the task is rejected (lost) and hence not processed at all. When only i processors are fault-free (structure state i, $0<i<N$), the system behaves as described above if each occurrence of the word "processor" is replaced by "fault-free processor." Upon failure of the system (state 0), processing ceases and any incoming task is rejected.

On closer inspection and in queueing theoretic terms (see [18], [19], for example), structure state i ($1 \leq i \leq N$) can be viewed as a queueing system with i servers (the fault-free processors), a finite queue of size L (the buffer), and a first-come-first-served queueing discipline. If, further, we assume that the processing times for each fault-free processor are independent and exponentially

distributed with parameter

μ = average processing rate (in the long run), (14)

then structure state i is an M/M/i/i+L queueing system. With this identification, a submodel of C in structure state i follows immediately by taking the internal states to be the set $Q_{I,i} = \{j \mid 0 \leq j \leq i+L\}$ where j = number of tasks in C. Letting $X_{I,i}$ denote the submodel in question, that is, the stochastic representation of an M/M/i/i+L queue with state set $Q_{I,i}$, it follows that $X_{I,i}$ is the "birth-death" Markov process given by the state-transition-rate diagram of Fig. 3. (In case i=0, we take $X_{I,0}$ to be a degenerate process consisting of a single absorbing state j=0.) The model parameters indicated in Fig. 3 are the task arrival rate α (2) associated with the environment X_E, the processing rate μ (14) of each processor, and the capacity L (7) of the buffer.

Composing the internal state submodels $X_{I,i}$ (Fig. 3) with the structure model X_R (Fig. 2), C can be modeled as a single Markov process X_C with state set $Q_C = \{(i,j) \mid i \in Q_R, j \in Q_{I,i}\}$ where, as defined above, i is the structure state of C and j is the number of tasks in C. The state-transition-rate diagram of the composite model X_C is shown in Fig. 4. For a structure state i such that $2 \leq i \leq N$, the transition to state (0,0) indicated at the far left of the diagram applies to each state (i,j) in the corresponding row of the diagram. X_C, together with the environment model X_E, thus constitute the base model of the system S = (C,E). However, with respect to the performance variable Y_S (see (1), (4)) we find that the relevant aspects of X_E have been incorporated in X_C, so that X_C can serve as the base model of S, i.e., $X_S = X_C$.

As a base model, X_C is similar in both its purpose and its appearance (Fig. 4) to the kind of "workload models" considered by Gay and Ketelsen [10]. One difference is that we make no assignment of "capacities" to the states of the model. Rather, the computational capacity of a given structural configuration is implied by certain transition rates, i.e., in structure state i, the maximum processing rate is $i\mu$ tasks/unit time. The major difference, however, is that the systems considered in [10] are repairable, resulting in irreducible Markov models where all states are recurrent non-null. The model of Fig. 4, on the other hand, has transient (non-recurrent) states; indeed, all the states of X_C are transient except for the absorbing state (0,0). This difference has a considerable impact on techniques that can be used to solve the model, as we discuss in the

section that follows.

III. MODEL SOLUTION

Since the performance variable Y_S (see (1),(4)) is continuous, a solution of performability requires solution of the probability distribution function (PDF) of Y_S. To this end and to simplify notation, let Y denote Y_S and let F_Y denote the PDF of Y, that is, $F_Y(y) = \text{Prob}[Y \leq y]$. Then, ideally, we would like to solve F_Y as an exact formulation of $F_Y(y)$, expressed in terms of y, t (the duration of utilization), and the parameters of the base model $X_S = X_C$ (Fig. 4). The parameters involved, including those derived from basic parameters, are summarized in Table 1. Such a formulation, however, would require (among other things) an exact, time-dependent solution of the state probabilities of the base model. Although this is possible, in principle, it is fraught with practical difficulties. Indeed, for even the simplest models of this sort, e.g., an M/M/1 queue, such a solution is far from trivial (see [18], pp. 73-78). On the other hand, if we are willing to settle for a good approximate solution, many of these difficulties may be circumvented.

Adopting the latter strategy, let us suppose the system is such that the utilization time and the average failure times of the resources are much larger than the average interarrival time of incoming tasks and the average processing time of a processor, i.e.,

$$t,\ 1/\lambda_p,\ 1/\lambda_b \gg 1/\alpha,\ 1/\mu. \tag{15}$$

Note that this situation will prevail in most computing system applications since the quantities on the left are usually multiples of hours while those on the right are typically fractions of seconds. For example, if t = 10 hours and $1/\alpha$ = 1 second then $\frac{t}{1/\alpha} = 36{,}000$. Assuming (15) (as we do throughout the remainder of the discussion), from the formulation of λ_i (see (12)), it follows that

$$t,\ 1/\lambda_i \gg 1/\alpha,\ 1/\mu.$$

Accordingly, for each structure state i ($1 \leq i \leq N$) let

$$W_t^i = \text{total time spent in state i during } [0,t] \tag{16}$$

and suppose that the system enters state i during utilization, i.e., $W_t^i > 0$. Then, with high probability, $W_t^i \gg 1/\alpha,\ 1/\mu$. In other words, the time spent in a structure state (if entered) is likely to be long compared to the intertransition times

among the internal states of that structure (see Fig. 4).

Thus, to a good first approximation, the internal state behavior in structure state i can be viewed as the long run, equilibrium behavior of the process $X_{L,i}$ (Fig. 3). More precisely, if we let S_i denote an M/M/i/i+L queueing system (the system modeled by $X_{L,i}$) and we let

$$r_i = \text{normalized average throughput rate of } S_i \text{ (in equilibruim)} \tag{17}$$

then r_i is the rate at which i contributes to the performance of S. Moreover, accounting for the fact that $r_0=0$, one can easily verify that

$$Y = \sum_{i=1}^{N} r_i \frac{W_t^i}{t} \tag{18}$$

where, by (18), $\frac{W_t^i}{t}$ is just the fraction of the period [0,t] that the system is in structure state i.

Mathematically, the performance variable Y is now expressed as a function of lower level variables r_i and W_t^i ($1 \le i \le N$). By their definitions, each variable r_i can be solved in terms of the equilibrium behavior of its corresponding queueing model $X_{L,i}$. As is well known (see [18], [19], for example), the equilibrium distribution of each r_i is deterministic (i.e., r_i assumes a constant value with probability 1) whence Y reduces to a linear combination of the (dependent) variables $W_t^1, W_t^2, \ldots, W_t^N$. Accordingly, the first step is to obtain closed-form solutions of the equilibrium rates $r_1, r_2, \ldots, r_N$.

Equilibrium Solutions

As above, let S_i denote an M/M/i/K queue, with K=i+L, and let p_K denote the equilibrium probability of finding S_i in state K (full queue). Then it is known (see [19], for example) that the average arrival rate (in equilibrium) of tasks that actually enter the system is $\alpha(1-p_K)$. Since this coincides with the average throughput rate of S_i (in equilibrium), on normalizing by the arrival rate α we conclude that

$$r_i = 1-p_K. \tag{19}$$

The general solution of p_K is known (see [19], Appendix C, Table 8, for example) and can be expressed as a function of i and the model parameters L=K−i, α and μ. Moreover, the dependence on α and μ is only through their ratio

$$u = \frac{\alpha}{\mu} \quad , \tag{20}$$

the so-called "traffic intensity." By (19), these remarks apply as well to r_i which, in a general form, can be expressed as follows:

$$r_i = \begin{cases} \dfrac{\left[1-\dfrac{u}{i}\right]\displaystyle\sum_{n=0}^{i}\frac{u^n}{n!} + \frac{u^i}{i!}\left[\frac{u}{i}\right] - \frac{i^i}{i!}\left[\frac{u}{i}\right]^{L+i}}{\left[1-\dfrac{u}{i}\right]\displaystyle\sum_{n=0}^{i}\frac{u^n}{n!} + \frac{u^i}{i!}\left[\frac{u}{i}\right] - \frac{i^i}{i!}\left[\frac{u}{i}\right]^{L+i+1}} \quad , & \text{if } u \neq i \\ \dfrac{\displaystyle\sum_{n=0}^{i}\frac{i^n}{n!} + \frac{i^i}{i!}\left[L-1\right]}{\displaystyle\sum_{n=0}^{i}\frac{i^n}{n!} + \frac{i^i}{i!}L} \quad , & \text{if } u = i \end{cases} \tag{21}$$

From these expressions it follows that, for fixed L, the normalized average throughput rate r_i is a monotonically decreasing function of u where, in the limit, $r_i \to 0$ as $u \to \infty$. On the other hand, for fixed u, r_i is a monotonically increasing function of the buffer capacity L, as one would expect, since the larger the buffer, the less chance there is of losing a task. Although we could examine the functional properties of r_i in greater detail, they are well understood (by people familiar with queueing systems) and, for the purpose of the development that follows, the above observations should suffice.

Solution of Performability

Since the variables r_i assume constant values for fixed values of the base model parameters, by (18) the performance variable Y can be expressed as a linear combination of lower level random variables, viz.

$$Y = \frac{1}{t}\sum_{i=1}^{N} r_i W_t^i \tag{22}$$

where W_t^i (16) is the total time spent in structure state i during [0,t]. Moreover, as the variables W_t^i depend only on the structure model X_R (Fig. 2), X_R can serve as the base model for the remaining part of the solution process. Accordingly, if equation (22) is extended to include state i = 0 where, trivially, $r_0 = 0$ (see (17)), the equilibrium solutions $r_0, r_1, \ldots, r_N$ may be thought of as "yield rates"

assigned to states 0, 1, ..., N, respectively. In other words, the r_i constitute a "reward structure" (see [20]) for the Markov process X_R. To the best of our knowledge, however, the analysis of reward models has dealt exclusively with the solution of expected rewards, e.g., for the variable in question, the expected value E[Y] of Y. Performability evaluation, on the other hand, requires a complete probabilistic description of Y, as provided by its PDF F_Y.

At this point, however, we find that a probabilistic characterization is difficult to obtain since the random variables $W_t^1, W_t^2, \cdots, W_t^N$ are (statistically) dependent. This is due to the fact that the combined times spent in states 1, 2,..., N cannot exceed t. Thus, for example, $\text{Prob}[W_t^{N-1} > 0 \mid W_t^N = t] = 0$ whereas $\text{Prob}[W_t^{N-1} > 0 \mid 0 < W_t^N < t] = c_N$ (see (13)), thereby demonstrating the dependence between W_t^N and W_t^{N-1}. In general, whenever performance is defined with respect to a bounded utilization period, such dependencies are likely to exist among variables that are closely related to the performance variable.

To circumvent this difficulty, a possible approach (which, in retrospect, appears to be the key to solving such problems) is to search for a lower level model which, at the expense of a more complex relation to Y, has a simpler probabilistic description. For the situation in question, we obtain such a model by considering the times spent in structure states 1, 2,..., N over the entire unbounded interval $[0,\infty)$. More precisely, we take the lower level model to be the sequence of variables $V = (V_1, V_2, \ldots, V_N)$ where

$$V_i = \lim_{t\to\infty} W_t^i = \text{time spent in state } i \text{ during } [0,\infty). \tag{23}$$

Although this model is no less "abstract" than that described by the sequence $(W_t^1, W_t^2, \ldots, W_t^N)$, it should be clear that it contains more information, thereby permitting Y to be formulated as function γ of V. To establish the specific nature of γ (referred to in [12] as a "capability function"), let $v = (v_1, v_2, \ldots, v_N)$ denote a value of V and, for notational convenience, let σ_j denote the sum

$$\sigma_j = \sum_{i=j}^{N} v_i \;;\; 1 \le j \le N.$$

Then it is relatively easy to verify that

$$\gamma(v) = \begin{cases} \frac{1}{t}\sum_{i=1}^{N} r_i v_i & \text{if } \sigma_1 \le t \\ r_j + \frac{1}{t}\sum_{i=j+1}^{N} (r_i - r_j)v_i & \text{if } \sigma_{j+1} \le t,\ \sigma_j > t \\ r_N & \text{if } \sigma_N > t\ . \end{cases} \tag{24}$$

At the cost of a more complicated capability function, we are now at a level where a probabilistic characterization is easier to obtain. This, in turn, can provide the solution we seek, since $Y = \gamma(V)$ and hence $F_Y(y)=\text{Prob}[\gamma(V)\le y]$. Consequently, if we let $B_y=\{b \mid b\le y\}$ then

$$F_Y(y) = \text{Prob}\,[V\varepsilon\gamma^{-1}(B_y)]. \tag{25}$$

To formulate these probabilities, we note first that, over the unbounded period $[0,\infty)$, a state trajectory (sample path) of X_R (see Fig.2) will, with probability 1, pass through a finite sequence of distinct states, beginning in some initial state k and terminating in the absorbing state 0. For each state $i > 0$ that is visited, the variable V_i is thus the time of a single "sojourn" is state i. Moreover, since X_R is a Markov process, it is known (see [21], for example) that these sojourn times are exponentially distributed and are conditionally independent, given the sequence of states that are visited.

With these observations, the solution of F_Y can be conveniently decomposed by considering the conditional PDF of Y with respect to the random variable

U = sequence of states (excluding 0) visited during $[0,\infty)$.

More specifically, by the transition structure of X_R, if a trajectory begins in state k where $k > 0$ and ends in state ℓ (prior to entering state 0) then $\ell \le k$ and

$U = (k,k-1,\ldots,\ell)$.

If a trajectory begins in state 0 then no states (other than 0) are visited during $[0,\infty)$, in which case

$U = \Lambda$ (the null sequence).

Thus, for an N-processor system, there are $\frac{N(N+1)}{2} + 1$ possible values of U. Accordingly, if we let u denote a value of U and let

$F_{Y|U}$ = conditional PDF of Y given U=u

then $F_Y(y)$ may be expressed as

$$F_Y(y) = \sum_u F_{Y|U}(y|u)\text{Prob}[U=u] \tag{26}$$

where the sum is taken over all possible values of U. Moreover, for a given u, the terms $F_{Y|U}(y|u)$ and Prob [U = u] can be solved as follows.

Regarding $F_{Y|U}(y|u)$, if we let $C_y=\gamma^{-1}(B_y)$ then, in view of (25), when Y is conditioned by U=u we have

$$F_{Y|U}(y|u) = \text{Prob}[V \in C_y|U = u].$$

This says, in turn, that $F_{Y|U}(y|u)$ can be solved by integrating the conditional joint probability density function (pdf) of V given U = u over the region C_y. More precisely, if we let

$f_{V|U}$ = conditional joint pdf of V given U=u

then

$$F_{Y|U}(y|u) = \int_{C_y} \cdots \int f_{V|U}(v|u)\, dv_1\, dv_2 \cdots dv_N . \tag{27}$$

The formulation of $f_{V|U}(v|u)$ is straightforward, due to the independence of the sojourn times V_i corresponding to states in the sequence u. Given u, for each state $i \in u$ (meaning, with a slight abuse of notation, that i appears in the sequence u), we know that V_i is exponentially distributed with parameter λ_i (see (12) and Fig. 2); if $i \notin u$ then, with probability 1, $V_i = 0$. Consequently, by the independence of the V_i, if u is nonnull then

$$f_{V|U}(v|u) = \begin{cases} \prod_{i \in u} \lambda_i e^{-\lambda_i v_i} & \text{if } v_j = 0, \text{ for all } j \notin u \\ 0 \text{ if } v_j > 0, \text{ for some } j \notin u. \end{cases} \tag{28}$$

In case $u = \Lambda$ (the null sequence) the formulation is trivial, i.e.,

$$f_{V|U}(v|\Lambda) = \begin{cases} 1 & \text{if } v_1=v_2=\cdots=v_N=0 \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Determining the multiple integral of equation (27), on the other hand, is generally quite difficult, due to the nonlinear form of the capability function γ (see (24)). Details of this process are illustrated, for the case N = 2, in the section that follows.

As for the second product term in equation (26), the solution of Prob[U = u] is immediate by inspection of the transition-rate diagram of X_R (Fig. 2). Given a state sequence u, u may be viewed a trajectory of the "imbedded" discrete-time Markov process $\bar{X}$ obtained by sampling X_R each time it changes state. Moreover, by inspection of X_R, if $i \geq 2$, then $\text{Prob}[\bar{X}_{n+1} = i-1 \,|\, \bar{X}_n = i] = c_i$ (see (13)) and $\text{Prob}[\bar{X}_{n+1} = 0 \,|\, \bar{X}_n = i] = 1 - c_i$; if $i = 1$, $\text{Prob}[\bar{X}_{n+1} = 0 \,|\, \bar{X}_n = 1] = 1$. Accordingly, if we let $\{p_i \,|\, 0 \leq i \leq N\}$ denote the initial state probability distribution of X_R, i.e.,

$$p_i = \text{Prob}[X_{R,0} = i]$$

then, for a nonnull sequence $u = (k,k-1,\ldots,l)$

$$\text{Prob}[U = u] = \begin{cases} p_k c_k c_{k-1} \cdots c_{l+1}(1 - c_l) & \text{if } k > l \geq 2 \\ p_k c_k c_{k-1} \cdots c_2 & \text{if } k > l = 1 \\ p_k(1 - c_k) & \text{if } k = l \geq 2 \\ p_1 & \text{if } k = l = 1 \,. \end{cases} \tag{30}$$

In case u is the null sequence, the corresponding trajectory must initially be in state 0; hence

$$\text{Prob}[U = \Lambda] = p_0. \tag{31}$$

This completes the description of the solution procedure which, in summary, involves the following steps:

1) For each structure state i, apply (21) to determine the equilibrium solution r_i of the normalized average throughput rate in state i.
2) For each state sequence u, apply (28), (29) to determine the conditional joint pdf of V given U = u and then apply (27) to determine the PDF of Y given U = u.
3) For each state sequence u, apply (30), (31) to determine the probability that U = u.
4) Combining the results of 2) and 3), apply (26) to determine the PDF F_Y of the performance variable Y.

Dual-Processor Example

To illustrate this procedure and, particularly, the kind of solutions it is capable of producing, let us consider the case of a buffered dual-processor (N = 2).

Step 1)

On substituting the values i=1 and i=2, respectively, in the general equilibrium solution (21) and after appropriate simplifications, r_1 and r_2 have the following solutions:

$$r_1 = \begin{cases} \dfrac{1-u^{L+1}}{1-u^{L+2}} & \text{if } u \neq 1 \\ \dfrac{L+1}{L+2} & \text{if } u=1 \end{cases} \tag{32}$$

$$r_2 = \begin{cases} \dfrac{1+\frac{u}{2}-2(\frac{u}{2})^{L+2}}{1+\frac{u}{2}-2(\frac{u}{2})^{L+3}} & \text{if } u \neq 2 \\ \dfrac{2L+3}{2L+5} & \text{if } u = 2 \; . \end{cases} \tag{33}$$

Step 2)

When N = 2, there are four state sequences to consider: $u_1 = (2,1)$, $u_2 = (2)$, $u_3 = (1)$, and $u_4 = \Lambda$. Interpreting these sequences, if a state trajectory (of X_R) has sequence u_1 then the system is initially fault-free and, during $[0,\infty)$, recovers from a single processor fault before eventually failing. If the sequence is u_2, the system is initially fault-free but fails on the first occurrence of a processor or buffer fault. u_3 says that one processor is initially faulty; u_4 says that the system failed prior to utilization. Applying Step 2) with respect to sequence u_1 (the remaining cases are simpler and we omit their illustration), by (28) we have

$$f_{V|U}(v|u_1) = \lambda_1 e^{-\lambda_1 v_1} \lambda_2 e^{-\lambda_2 v_2} \; .$$

To subsequently obtain its integral (see (27)) over the region $C_y = \{v|\gamma(v) \leq y\}$, it is necessary to characterize C_y for various ranges of y so as to determine the specific limits of integration. This is done by specializing γ (24) to the case in point (N = 2) and examining the boundary $\gamma^{-1}(y)$ that delimits the region C_y. Thus, for example, if y is in the range $r_1 \leq y < r_2$, then C_y is the region of the v_1-v_2 plane depicted in Fig. 5. For convenience in stating the resulting solution, let ν_1 denote the quantity

$$\nu_1 = \frac{\lambda_1}{r_1} \; . \tag{34}$$

Note that when λ_i (12) and r_i (32), (33) are fully expressed, ν_i is a function of i and the base model parameters λ_p, λ_b, L, α, and μ (see Table 1). Then, for the instance where y is in the range $r_1 \leq y < r_2$, integration over C_y (see Fig. 5) yields the solution

$$F_{Y|U}(y|u_1) = 1 - \left[e^{-\nu_2 t y} + \frac{\nu_2 \, e^{-\nu_1 t y} \, e^{\frac{-(\nu_2-\nu_1)tr_2(y-r_1)}{(r_2-r_1)}} - e^{-\nu_2 t y}}{(\nu_2-\nu_1)} \right].$$

Solutions with respect to other ranges of y and other values of U are obtained in a like manner.

Step 3)

By the definitions of $u_1, \ldots, u_4$ and on applying (30), (31), we have

$\text{Prob}[U = u_1] = p_2 c_2$.

$\text{Prob}[U = u_2] = p_2(1 - c_2)$.

$\text{Prob}[U = u_3] = p_1$.

$\text{Prob}[U = u_4] = p_0 = 1 - (p_1 + p_2)$.

Step 4)

Applying equation (26) to the results obtained in steps 2) and 3), we obtain a closed-form solution of F_Y, expressed in terms of y, r_i (see (32), (33)) and ν_i (see (34)). This solution is displayed in Table 2, thereby completing the procedure.

Given F_Y, we thus obtain a closed-form solution of the performability p_S for intervals of the form $B_y = \{b \mid b \leq y\}$, i.e.,

$$p_S(B_y) = F_Y(y).$$

To get a clearer picture of what this solution looks like, Figs. 6 and 7 display plots of $p_S(B_y) = F_Y(y)$ as a function of y for various choices of t and the base model parameters. Fig. 6 considers the system where t = 240 (10 days), $u = \frac{\alpha}{\mu} = 1.5$, $\lambda_p = 10^{-3}$, $\lambda_b = 10^{-4}$, $c_p = 0.999$, $p_2 = 0.9$, $p_1 = 0.09$, and $p_0 = 0.01$. The figure furnishes several plots showing how $F_Y(y)$ varies as L ranges from 0 to 10 in steps of 2. Fig. 7 is similar to Fig. 6 except that $p_2 = 1.0$ (whence

$p_1 = p_0 = 0$), i.e., the system is initially fault-free.

Solution with respect to an arbitrary measurable set B of accomplishment levels can then be formulated via integration (see [12], eqn. (9)). In practice, however, the sets B are typically intervals, in which case the performability values are provided directly by F_Y. For example, if one is interested in the system's ability to perform within specified limits b_0 and b_1 ($b_0<b_1$) then $B=\{b \mid b_0<b\leq b_1\}$, whence $p_S(B)=F_Y(b_1)-F_Y(b_0)$. Another example, and one that arises even more frequently in practice, is the ability of the system to perform above a specified "performance threshold" b_0. In this case, $B=\{b \mid b>b_0\}$ and $p_S(B)=1-F_Y(b_0)$.

Application of the solution

To illustrate an application of the dual-processor solution (Table 2), let us suppose the designer wants to maximize the probability that the normalized average throughput rate exceeds a specified fraction b_0. In other words, as discussed above, the accomplishment set in question is $B=\{b \mid b>b_0\}$ and we want to maximize the probability $p_S(B)=1-F_Y(b_0)$. Suppose further that the design choice is the value of the buffer capacity L, which is interesting since this choice can influence performance and reliability in a compensating manner. Were performance the only issue, then L should be made as large as possible (subject to other practical constraints such as cost) since the larger the buffer, the higher the normalized average throughput rate (see (21)). On the other hand, if reliability were the only issue, then no buffer at all (L = 0) is the best choice since it will minimize the probability of system failure. Realistically, however, both performance and reliability are issues and, when considered simultaneously, we find that the performability (relative to a specified set B) can be optimized by an appropriate choice of L.

For example, suppose $B = \{b \mid b > 0.75\}$, i.e., the system S performs within B if the normalized average throughput rate is greater than 0.75. Then, for various choices of the buffer stage failure rate λ_b (the remaining parameter values are as in Fig. 7), the variation of $p_S(B)$ as a function of buffer capacity L is displayed in Fig. 8. In the case of a perfectly reliable buffer ($\lambda_b = 0$), we note that performability is a monotonically increasing function of the buffer capacity L, as one would expect. On the other hand, when buffer stages fail at a nonzero rate, we see that L can be chosen so as to maximize performability where, the

higher the failure rate, the lower the optimum value of L.

This is but one example of how such a closed-form solution of performability might be applied. Indeed, for the solution in question (Table 2), we have only begun to investigate its implications. Therefore, we intend to continue our exploration of various properties of this solution. We also want to investigate how the modeling and solution techniques discussed herein might be extended so as to apply to a more general class of systems.

REFERENCES

[1] D. Ferrari, *Computer Systems Performance Evaluation*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[2] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Reading, MA: Addison-Wesely, 1978.

[3] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*. New York: John Wiley & Sons, 1976.

[4] W. G. Bouricius, W. C. Carter, and P. R. Schneider, "Reliability modeling techniques for self-repairing computer systems," *Proc. ACM 1969 Annual Conf.*, pp. 295-309, Aug. 1969.

[5] J. C. Laprie, "Reliability and availability of repairable structures," in *Proc. 1975 Int'l Symp. on Fault-Tolerant Computing*, Paris, France, pp. 87-92, June 1975.

[6] Y.-W. Ng and A. Avizienis, "A reliability model for gracefully degrading and repairable fault-tolerant systems," in *Proc. 1977 Int'l Symp. on Fault-Tolerant Computing*, Los Angeles, CA, pp. 22-28, June 1977.

[7] A. Costes, C. Landrault, and J. C. Laprie, "Reliability and availability models for maintained systems featuring hardware failures and design faults," *IEEE Trans. Comput.*, vol. C-27, pp. 548-560, June 1978.

[8] J. F. Meyer, "On evaluating the performability of degradable computing systems," in *Proc. 1978 Int'l Symp. on Fault-Tolerant Computing*, Toulouse, France, pp. 44-49, June 1978.

[9] M. D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Comput.*, vol. C-27, pp. 540-547, June 1978.

[10] F. A. Gay and M. L. Ketelsen, "Performance evaluation for gracefully degrading systems," in *Proc. 1979 Int'l Symp. on Fault-Tolerant Computing*, Madison, Wisconsin, pp. 51-58, June, 1979.

[11] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer," *IEEE Trans. Comput.*, vol. C-29, pp. 501-509, June 1980.

[12] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vol. C-29, pp. 720-731, Aug. 1980.

[13] X. Castillo and D. P. Siewiorek, "A performance-reliability model for computing systems," in *Proc. 1980 Int'l Symp. Fault-Tolerant Computing*, Kyoto, Japan, pp. 187-192, Oct. 1980.

[14] J. F. Meyer and L. T. Wu, "Evaluation of computing systems using functionals of a Markov process," in *Proc. 14th Annual Hawaii Int'l Conf. on System Sciences*, Honolulu, HI, pp. 74-83, Jan. 1981.

[15] X. Castillo and D. P. Siewiorek, "Workload, performance, and reliability of digital computing systems," in *Proc. 1981 Int'l Symp. Fault-Tolerant Computing*, Portland, ME, pp. 84-89, June 1981.

[16] R. Huslende, "A combined evaluation of performance and reliability for degradable systems," in *ACM/SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Las Vegas, NV, pp. 157-164, Sept. 1981.

[17] J. F. Meyer, "Closed-form solutions of performability," in *Proc. 1981 Int'l Symp. Fault-Tolerant Computing*, Portland, ME, pp. 66-71, June 1981.

[18] L. Kleinrock, *Queueing Systems, Volume I: Theory*. New York: John Wiley & Sons, 1975.

[19] A. O. Allen, *Probability, Statistics, and Queueing Theory—With Applications*. New York: Academic Press, 1978.

[20] R. A. Howard, *Dynamic Probabilisitc Systems, Vol. II: Semi-Markov and Decision Processes*. New York: John Wiley, 1971.

[21] E. Cinlar, *Introduction to Stochastic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

CAPTIONS

Figure 1. Block diagram of C.

Figure 2. State-transition-rate diagram of X_R.

Figure 3. State-transition-rate diagram of $X_{I,i}$.

Figure 4. State-transition-rate diagram of X_C.

Figure 5. C_y for y in the range $r_1 \leq y < r_2$.

Figure 6. Plot of $p_S(B_y) = F_Y(y)$ as a function of y for the indicated choices of t and the base model parameters.

Figure 7. Plot of $p_S(B_y) = F_Y(y)$ as a function of y for the indicated choices of t and the base model parameters.

Figure 8. $p_S(B)$ as a function of the buffer capacity L.

Table 1. Base model parameters.

Table 2. Closed-form solution of F_y.

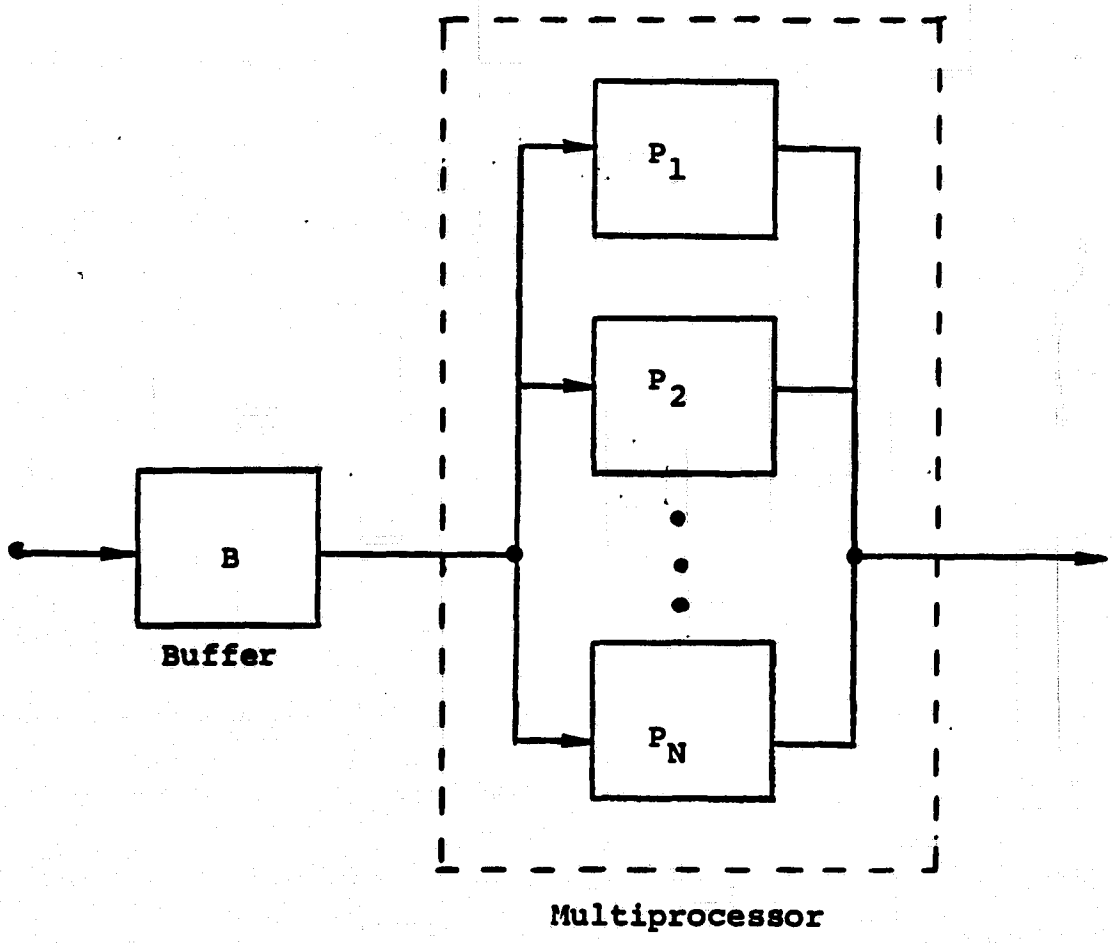

Fig.1. Block diagram of C.

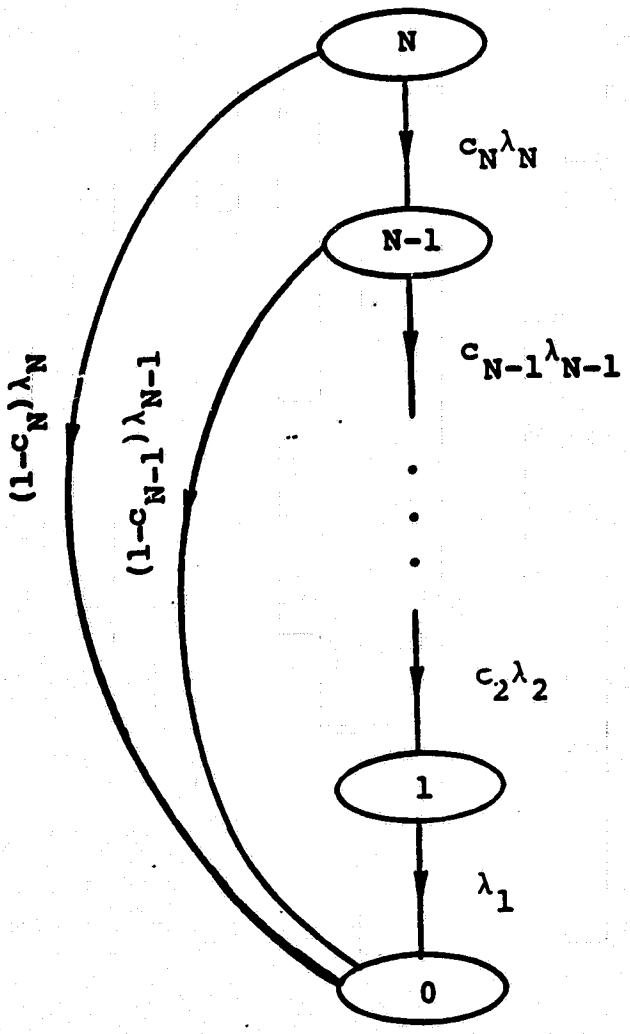

Fig.2. State-transition-rate diagram of X_R.

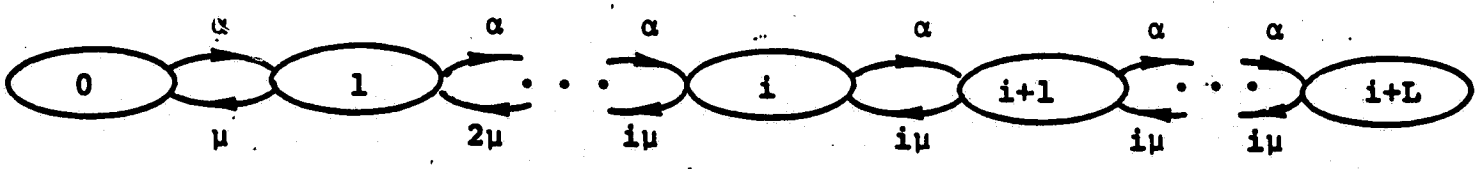

Fig.3. State-transition-rate diagram of $X_{I,i}$.

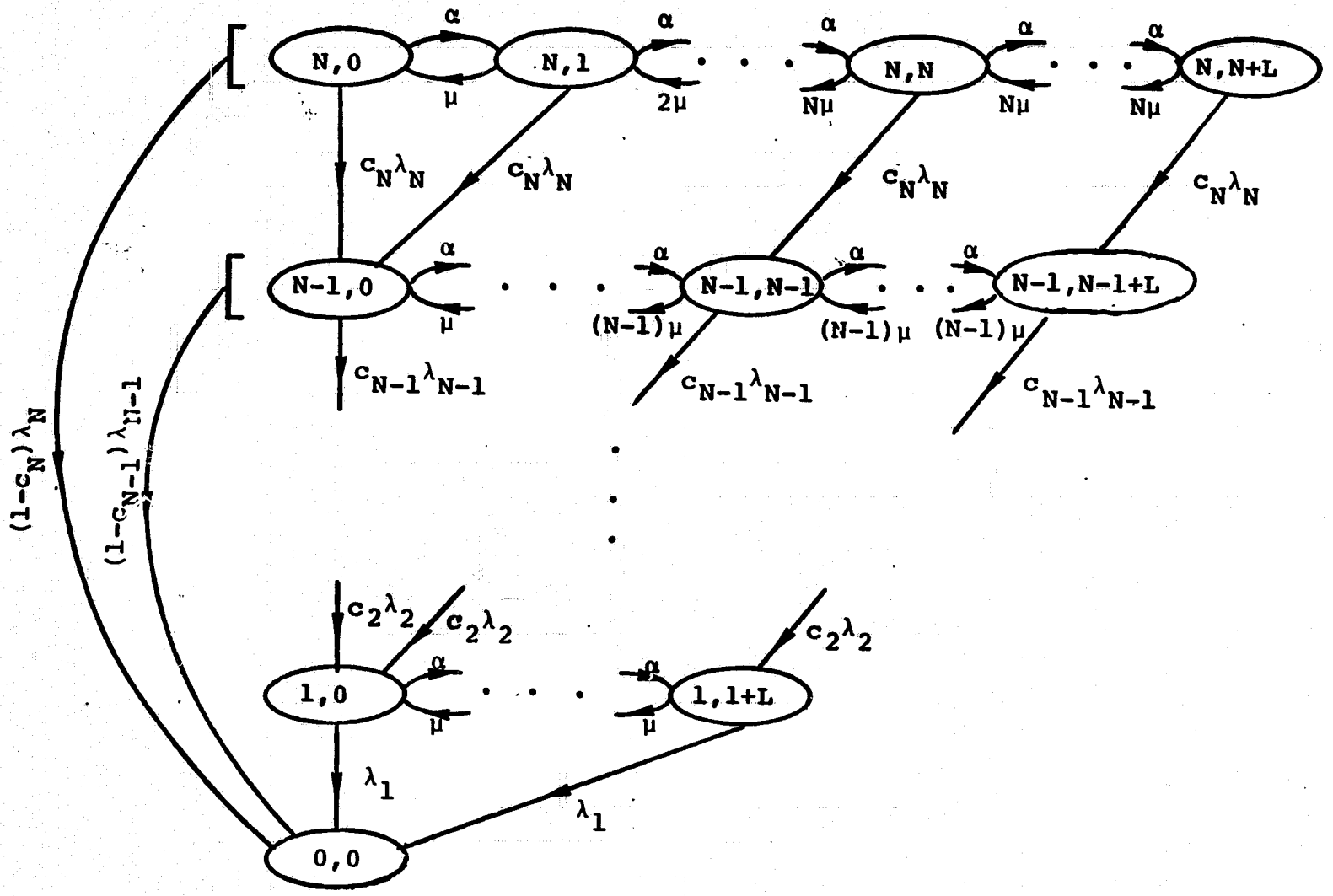

Fig.4. State-transition-rate diagram of X_C.

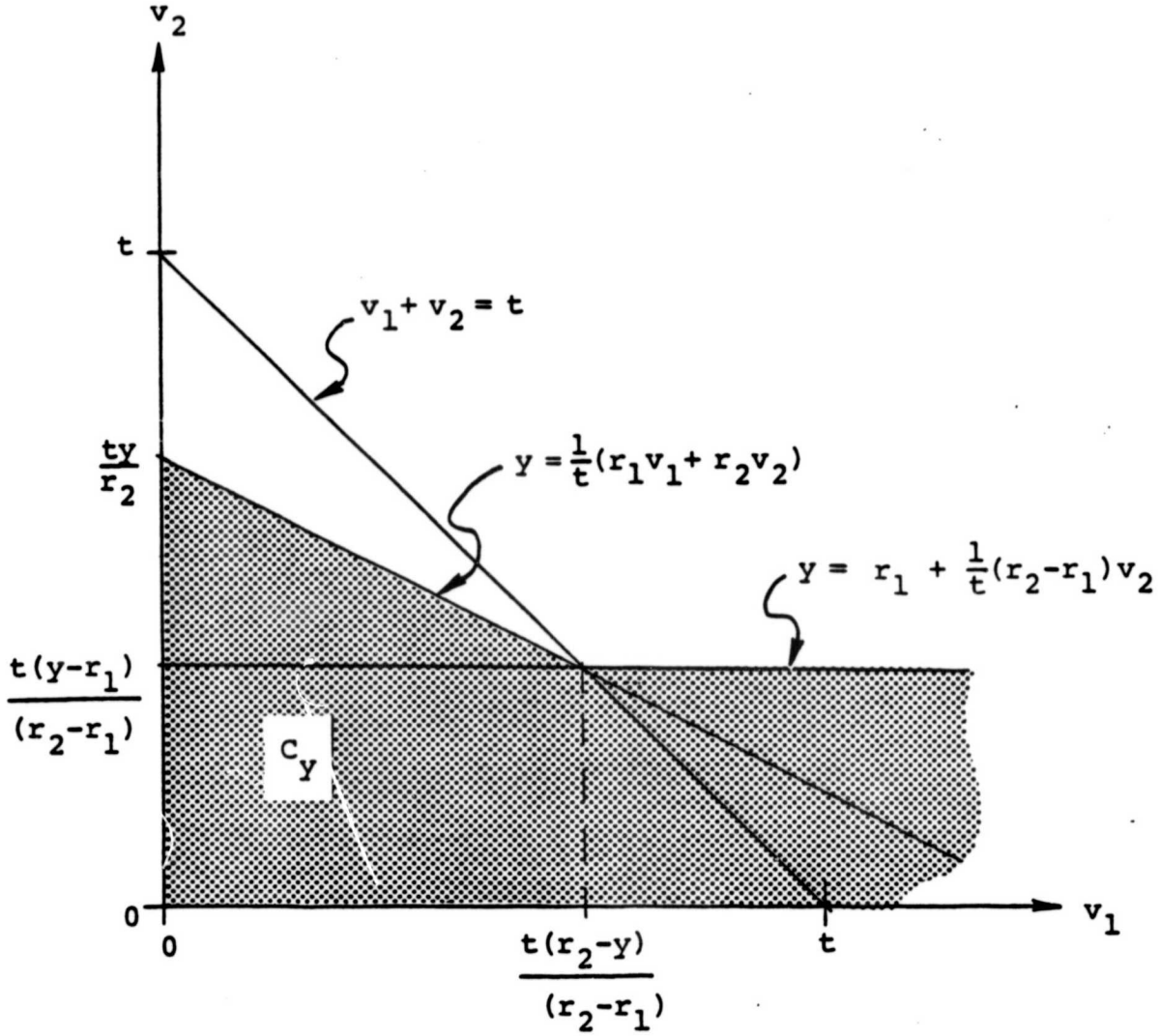

Fig.5 C_y for y in the range $r_1 \leq y < r_2$.

$t = 240$

$u = \frac{\alpha}{\mu} = 1.5$

$\lambda_P = 10^{-3}$

$\lambda_b = 10^{-4}$

$c_P = 0.999$

$p_2 = 0.90$

$p_1 = 0.09$

$p_0 = 0.01$

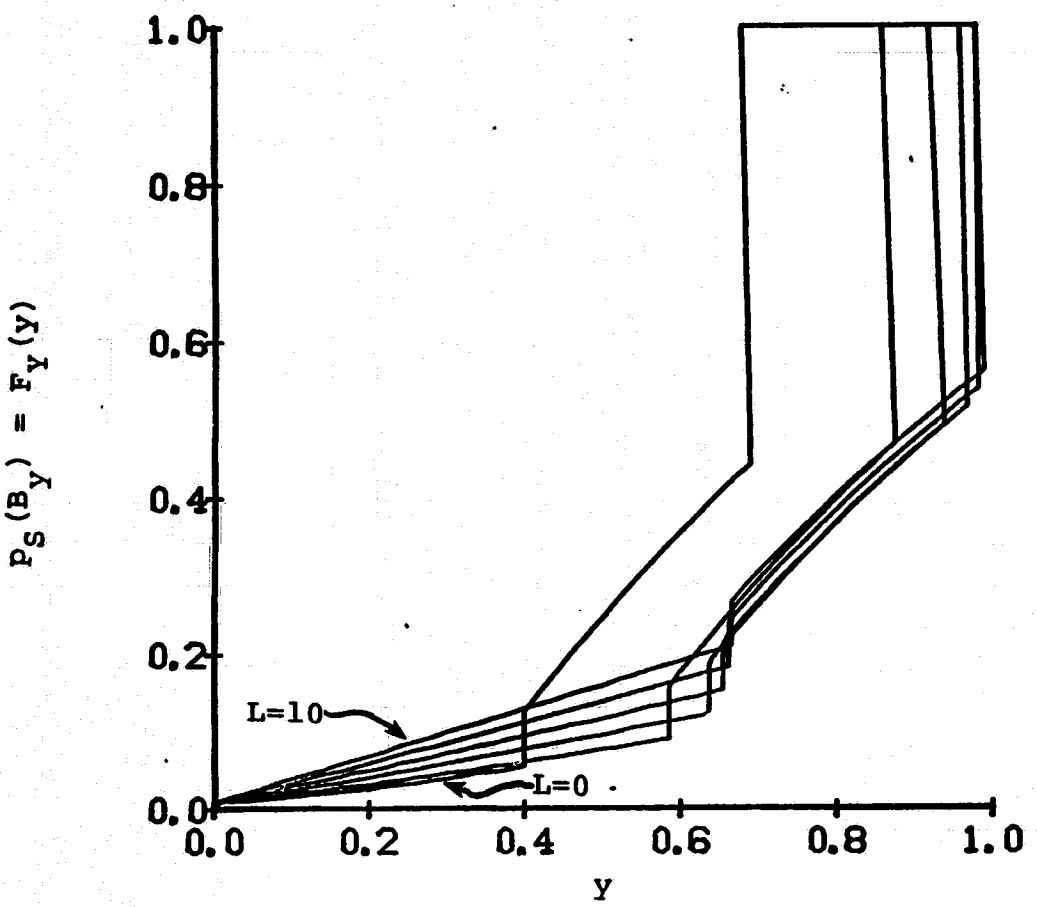

Fig. 6. Plot of $p_S(B_y)=F_Y(y)$ as a function of y for the indicated choices of t and the base model parameters.

$t = 240$

$u = \frac{\alpha}{\mu} = 1.5$

$\lambda_p = 10^{-3}$

$\lambda_b = 10^{-4}$

$c_p = 0.999$

$p_2 = 1.00$

$p_1 = 0.00$

$p_0 = 0.00$

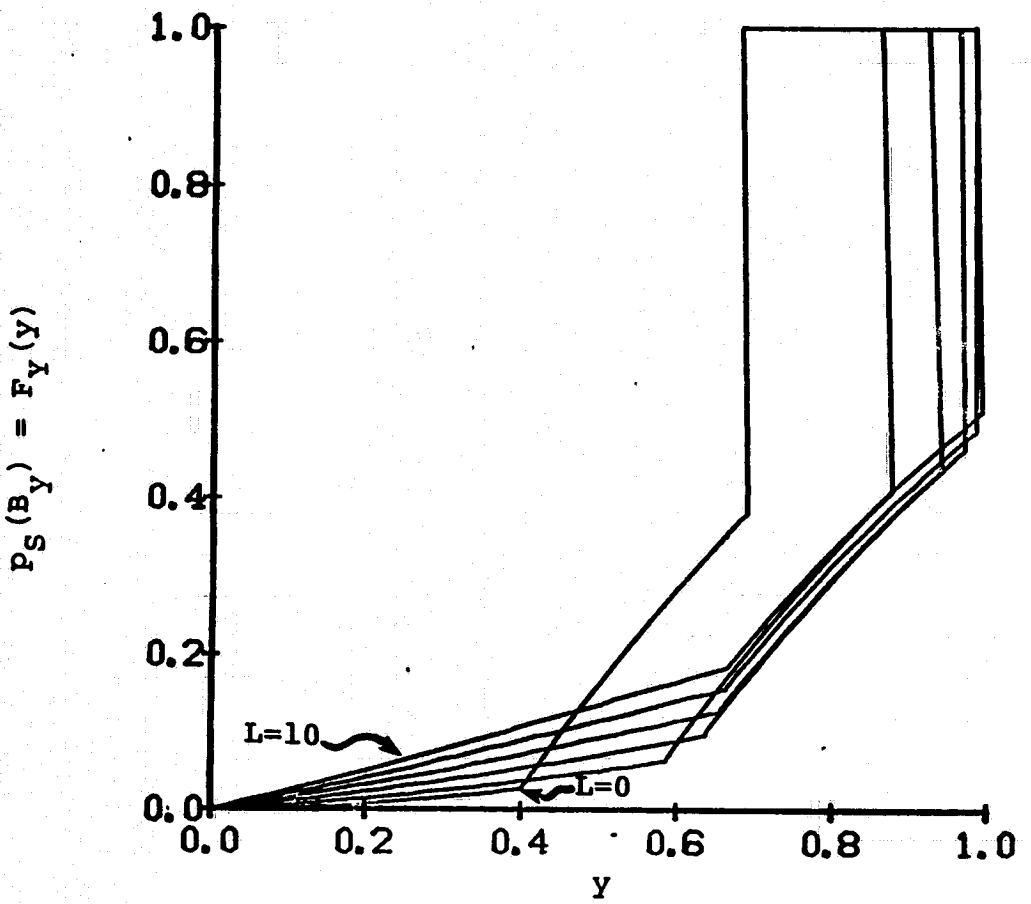

Fig. 7. Plot of $p_S(B_y)=F_Y(y)$ as a function of y for the indicated choices of t and the base model parameters.

$B = \{b | b > 0.75\}$

$t = 240$

$u = \frac{\alpha}{\mu} = 1.5$

$\lambda_p = 10^{-3}$

$c_p = 0.999$

$p_2 = 1.00$

$p_1 = 0.00$

$p_0 = 0.00$

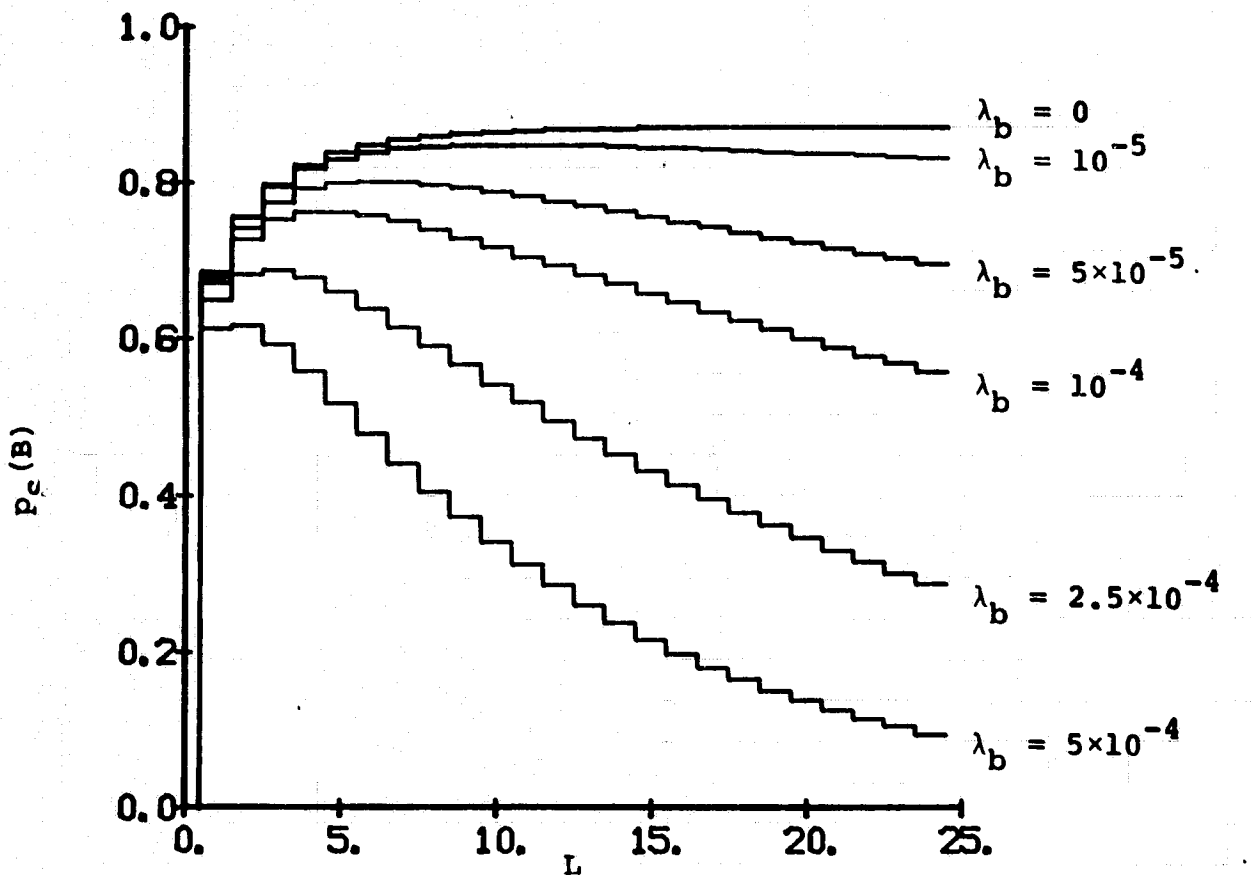

Fig. 8. $p_S(B)$ as a function of the buffer capacity L.

		SYMBOL	NAME	DEFINITION
ENVIRON-MENT		α	task arrival rate	(5)
COMPUTER	BASIC	N	number of processors	(6)
		L	buffer capacity	(7)
		λ_p	processor failure rate	(8)
		c_p	processor coverage	(9)
		μ	processor processing rate	(14)
		λ_b	buffer stage failure rate	(10)
	DERIVED	λ_B	buffer failure rate	(11)
		λ_i	transition rate from structure state i	(12)
		c_i	coverage in structure state i	(13)

Table 1. Base model parameters.

1) $0 \leq y < r_1$

$$F_Y(y) = 1 - p_2 \left[e^{-\nu_2 t y} + \frac{c_2 \nu_2 \left(e^{-\nu_1 t y} - e^{-\nu_2 t y} \right)}{(\nu_2 - \nu_1)} \right] - p_1 e^{-\nu_1 t y}$$

2) $r_1 \leq y < r_2$

$$F_Y(y) = 1 - p_2 \left[e^{-\nu_2 t y} + \frac{c_2 \nu_2 \left(e^{-\nu_1 t y} e^{-\frac{(\nu_2 - \nu_1) t r_2 (y - r_1)}{(r_2 - r_1)}} - e^{-\nu_2 t y} \right)}{(\nu_2 - \nu_1)} \right]$$

3) $r_2 \leq y$

$$F_Y(y) = 1$$

Table 2. Closed-form solution of F_Y.