014524-16-T
SEL 147

Closed-Form Solutions of Performability

J. F. Meyer

January 1981

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23365

G.E. Migneault, Technical Officer

NASA Grant NSG 1306

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
THE UNIVERSITY OF MICHIGAN, ANN ARBOR 48109

014524-16-T
SEL 147

CLOSED-FORM SOLUTIONS OF PERFORMABILITY

J. F. Meyer

Prepared For
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23365
G. E. Migneault, NASA Technical Officer
NASA Grant NSG 1306

J. F. Meyer, Principal Investigator
Department of Electrical and Computer Engineering
Systems Engineering Laboratory
The University of Michigan
Ann Arbor, Michigan 48109

January 1981

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. Prior work on performability models and solution methods has focused on the case where Y is discrete; in this paper we consider continuous-valued variables of the type usually addressed in performance evaluation (e.g., average throughput rate, average response time, etc.). The models used are similar to those employed in performance modeling (i.e., Markovian queueing models) but are extended so as to account for variations in structure due to faults. 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. To avoid known difficulties associated with exact transient solutions, we employ an approximate decomposition of the model, permitting certain submodels to be solved in equilibrium. These solutions are then incorporated in a model with fewer transient states and by solving the latter, we obtain a closed-form solution of the system's performability. In conclusion, some applications of this solution are discussed and illustrated, including an example of design optimization.

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]-[14], 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.

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, e.g., in the special case of reliability evaluation, A = {success, failure}. At the other extreme, performance may range over a continuum of values, e.g., A is the real number interval $[0,\infty)$ where a level $a \in A$ is the "throughput rate of S averaged over T." With respect to a designated performance variable Y_S, the "performability" of S is the probability measure induced by Y_S where, for any measurable set B of accomplishment levels ($B \subseteq A$),

$$p_S(B) = \text{probability that S performs at a level in B.}$$

(For more precise definitions of these concepts, the reader is referred to [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. In general (assuming Y_S is real-valued), this description is provided by the probability distribution function (PDF) of Y_S, i.e., the function F_{Y_S} where $F_{Y_S}(y) = \text{Prob}[Y_S \leq y]$. By the definition of performability, it follows that p_S is uniquely determined by F_{Y_S}; in particular, if $B_y = \{a \in A | a \leq y\}$ then $p_S(B_y)$ (the ability to perform within B_y) coincides with $F_{Y_S}(y)$. To solve F_{Y_S}, performability modeling calls for an appropriate representation of the total system S by a stochastic process X_S (the "base model" of S) so that each state trajectory (sample path) of X_S corresponds to a specific value of Y_S. (In the terminology of [8], [12], this correspondence is referred to as the "capability function" of S.) Typically, the modeling process will also involve the introduction of intermediate models between X_S and Y_S, so as to facilitate the solution of F_{Y_S}.

Prior work on the development of performability models and solution methods has dealt primarily with discrete performance variables ranging over a countable and typically finite set of accomplishment levels. 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.

In the discussion that follows, we demonstrate that closed-form solutions of performability can indeed be obtained for a continuous performance variable. The system we consider consists of a degradable multiprocessor with an input buffer (queue) for the temporary storage of computational tasks that arrive randomly at the input. The performance in question is the fraction of incoming tasks processed during utilization or, equivalently, the normalized average throughput rate. In constructing the base model of this system (Section II), we extend the kind of Markovian queueing models that are currently employed to evaluate the performance of a (fault-free) computer (see [1]-[3], for example). When so extended, these models are able to represent variations in structure, due to faults, as well as variations in internal state and environment. In solving the performability (Section III), our strategy is to lump states of the base model so that, within a lump, the model exhibits a steady-state behavior (to a close approximation). This permits decomposition of the solution into an equilibrium (steady-state) part and a transient part. The equilibrium part employs known solutions from queueing theory; the transient part is more difficult and calls for an approach which, to the best of our knowledge, is new. Here, through a hierarchical decomposition of the capability function and an appropriate partitioning of the accomplish-

ment set, we are able to obtain the desired solution.

II. MODEL CONSTRUCTION

The system we evaluate 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 ($N \geq 2$) 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 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 given once we specify the performance variable in question.

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 and D_t denote the random variables:

A_t = number of tasks that arrive during [0,t] (1)

D_t = number of tasks that are processed during [0,t]. (2)

Then, relative to the utilization period T = [0,t], we take the performance of S to be the random variable

$$Y_S = \frac{D_t}{A_t} = \text{fraction of arrived tasks processed during } T. \tag{3}$$

(Note that $0 \leq Y_S \leq 1$.) Alternatively, if we let

$$\alpha_t = \frac{A_t}{t} = \text{average arrival rate during } [0,t] \tag{4}$$

$$\delta_t = \frac{D_t}{t} = \text{average throughput rate of S during } [0,t] \tag{5}$$

then

$$\begin{aligned} Y_S &= \frac{D_t}{A_t} = \frac{D_t/t}{A_t/t} \\ &= \frac{\delta_t}{\alpha_t} = \frac{\text{average throughput rate of S during } T}{\text{average arrival rate during } T}. \end{aligned} \tag{6}$$

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 PDF 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 earlier (see (1)), 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

$$X_E = \{A_t | t \in [0,\infty)\} \tag{7}$$

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

α = average arrival rate (in the long run), (8)

that is, $\alpha = \lim_{t \to \infty} \alpha_t$ (see(4)), then X_E is uniquely determined by α. More precisely, in the terminology of Markov processes, X_E is a special "pure birth" process where the transition rate between each pair of successive states is α.

Computer Model

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

N = number of processors ($N \geq 2$) (9)

L = storage capacity of the buffer ($L \geq 0$). (10)

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 processor 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. 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

$Q_R = \{0,1,...,N\}$ (11)

with the following interpretation:

State	Fault-free resources
N	Buffer and N processors
N-1	Buffer and N-1 processors
.	.
.	.
.	.
i	Buffer and 1 processor
0	System failure .

Modeling how structure varies (probabilistically) as a function of time thus reduces to a standard problem typically 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

$$\lambda_p = \text{processor failure rate} \tag{12}$$

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

$$c_p = \text{probability of recovering from a processor fault.} \tag{13}$$

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,

$$\lambda_b = \text{stage failure rate.} \tag{14}$$

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

$$\begin{aligned} \lambda_B &= \text{buffer failure rate} \\ &= L\lambda_b. \end{aligned} \tag{15}$$

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 (11). 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,

$$\lambda_i = i\lambda_p + \lambda_B = i\lambda_p + L\lambda_b \ . \tag{16}$$

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 (13), it follows, via a simple conditional probability argument, that

$$c_i = \frac{i\lambda_p c_p}{i\lambda_p + L\lambda_b} = \frac{c_p}{1 + \frac{L\lambda_b}{i\lambda_p}} \ . \tag{17}$$

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. In case C is fault-free (structure state N), the system is presumed to behave as follows. Given that 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. Note that this last condition is the one which directly affects the performance $Y_S = D_t/A_t$ (see (3)) when C is fault-free, since $D_t < A_t$ if and only if tasks are lost during [0,t]. When only i processors are fault-free (structure state i, $i \geq 1$), the system behaves as described above if each occurrence of the word

"processor" is replaced by "fault-free processor." Upon failure of the system (structure state 0), processing ceases and any incoming task is rejected.

On closer inspection and in queueing theoretic terms (see [16], [17] 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 (the task scheduling discipline). If, further, we assume that the processing times for each fault-free processor are independent and exponentially distributed with parameter

$$\mu = \text{average processing rate (in the long run)}, \tag{18}$$

then structure state i is an instance of an M/M/m/K queueing system where

$$\begin{aligned} m &= i, \text{ the number of servers} \\ K &= i+L, \text{ the storage capacity of the system (servers queue plus).} \end{aligned} \tag{19}$$

(M/M denotes the fact that the interarrival times and service times are exponentially distributed.)

With this identification, a submodel of C in structure state i follows immediately by taking the internal states to be the set

$$Q_{I,i} = \{j | 0 \leq j \leq i+L\} \tag{20}$$

where

j = number of tasks in C,

that is, the number of tasks being processed plus the number of tasks stored in the buffer. (Thus, at the extremes, $j = 0$ says that all fault-free processors are idle and buffer B is empty; if $j = i + L$, all fault-free processors are busy and B is full.) 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. (Although $X_{I,i}$ is consistent with our interpretation even when $i = 0$, we will take $X_{I,0}$ to be a degenerate Markov process with a single absorbing state $j = 0$.) The model parameters indicated in Fig. 3 are the task arrival rate α (8) associated with the environment X_E, the processing rate μ (18) of each processor, and the capacity L (10) of the buffer. In other words, "births" correspond to tasks accepted by C and "deaths" to tasks completed by C. In particular, the zero acceptance rate in state $i = L$ reflects the fact that tasks are rejected when the buffer is full. Finally, if $j \geq i$ (in which case all fault-free processors are busy), the completion rate of $i\mu$ reflects the assumed (ideal) parallel processing capability of the multiprocessor.

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

$$Q_C = \{(i,j) | i \in Q_R,\ j \in Q_{I,i}\}$$

where, from the definitions of Q_R (11), and $Q_{I,i}$ (20), a state $q = (i,j)$ represents both the structure and internal state of C with

i = structural configuration of C,
j = 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.

The computer model X_C together with the environment model X_E (7) thus constitute the base model of the system S = (C,E). However, with respect to the performance variable Y_S (see (3), (6)) 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. Accordingly, we take the Markov process X_C to be the base model X_S and will subsequently refer to it by either name.

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. Also, in keeping with traditional usage (see [1]-[3]), we prefer to reserve the term "workload" for the external demands placed on the computer. A workload model is thus part of (and often coincides with) a model of the computer's environment, e.g., the arrival process X_E (7) is the "workload model" for the example in question. 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 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

As pointed out in the introductory remarks of Section I, solving a system's performability is tantamount to solving the probability distribution function (PDF) of the performance variable. To this end and to simplify notation for the system S in question, let Y denote Y_S (as specified in (3) or (6)) and let F_Y denote the PDF of Y, that is

$$F_Y(y) = \text{Prob}[Y \leq y]. \tag{21}$$

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 appears to be 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 [16], 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.

Choosing the latter approach, we note first that, in a form lying between equations (3) and (6), the performance variable $Y = Y_S$ can be expressed as

$$Y = \frac{D_t/t}{\alpha_t}. \tag{22}$$

To further decompose this expression, for each structure state i ($0 \leq i \leq N$), let us define a random variable D_T^i that represents t. contribution of i to D_t, that is,

$$D_t^i = \text{number of tasks processed in structure state i during } [0,t]. \tag{23}$$

Then, since no tasks are processed in structure state 0 (total failure), it follows that

$$D_t = \sum_{i=1}^{N} D_t^i. \tag{24}$$

If, further, we introduce the random variables

$$W_t^i = \text{total time spent in structure state i during } [0,t] \tag{25}$$

and let

$$\delta_t^i = D_t^i / W_t^i \qquad (26)$$

= average throughput rate in structure state i during [0,t].

then, from (24) and (26), we have

$$D_t = \sum_{i=1}^{N} \delta_t^i W_t^i.$$

Substituting this in (22), we can express Y as a function of lower level random variables, viz.

$$Y = \sum_{i=1}^{N} \frac{\delta_t^i \; W_t^i}{\alpha t} \qquad (27)$$

where, except for αt, each variable relates exclusively to a particular structure state.

In view of this formulation, suppose now that 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.,

$$t,\ 1/\lambda_p,\ 1/\lambda_b \gg 1/\alpha,\ 1/\mu. \qquad (28)$$

This case, and cases similar to it, 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 (28) (as we do throughout the remainder of the discussion), from the formulation of λ_i (see (16),

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

and hence, with high probability,

$$W_t^i \gg 1/\alpha,\ 1/\mu.$$

In other words, the time spent in a structure state is likely to be

long compared to the intertransition times among the internal states of that structure (see Fig. 4).

Therefore, 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 $X_{I,i}$ (Fig. 3). Moreover, these processes represent familiar systems in their own right where, letting S_i denote the system modeled by $X_{I,i}$, we recall (19) that

$$S_i = M/M/i/i+L \text{ queueing system .} \tag{29}$$

Accordingly, if we let

$$\delta^i = \text{average throughput rate of } S_i \text{ (in the long run)}$$

then, by the definition of δ_t^i (26),

$$\delta_t^i \cong \delta^i$$

and, since $t \gg 1/\alpha$,

$$\alpha_t \cong \lim_{s \to \infty} \alpha_s = \alpha.$$

Taking these approximations to be identities and substituting in (27), we find that Y can be approximated by the expression:

$$Y = \sum_{i=1}^{N} \frac{\delta^i \, W_t^i}{\alpha t}$$

If, further, we define

$$r_i = \frac{\delta^i}{\alpha} = \text{normalized average throughput rate of } S_i \text{ (in the long run),} \tag{30}$$

we obtain the following convenient formulation of the performance variable Y (normalized average throughput rate during [0,t]):

$$Y = \sum_{i=1}^{N} r_i \frac{W_t^i}{t} \quad . \tag{31}$$

Interpreting the terms of this formula, by definition (30) r_i approximates the normalized average throughput rate realized while the system is in structure state i; by definition (25), $\frac{W_t^i}{t}$ is just the fraction of time the system actually spends in state i. Thus equation (31) is intuitively quite plausible.

Mathematically, the performance variable Y is now expressed as a function of lower level variables r_i, and W_t^i ($1 \leq i \leq N$). By their definitions, each variable r_i can be solved in terms of the equilibrium behavior of its corresponding queueing model $X_{I,i}$. As is well known (see [16], [17], 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 defined above (29), the system S_i may be viewed as an ideal, fault-free version of S when the number of processors N is equal to i. With this view and on comparing (30) with (6), it follows that r_i is just the long run performance of this ideal, i-processor system. Reverting to our original definition of the performance variable (3), we thus obtain an alternative interpretation of r_i, namely

r_i = fraction of arrived tasks processed by S_i (in the long run). (32)

Moreover, since the fraction of arrived tasks that remain in S_i becomes negligible in the long run (there are at most i+L tasks in S_i), if we let

s_i = fraction of arrived tasks rejected by S_i (in the long run), (33)

then

$$r_i = 1 - s_i. \tag{34}$$

The above is a convenient formulation of r_i since the quantity s_i relates directly to the equilibrium state behavior of the queueing model $X_{I,i}$ (Fig. 3). Since $X_{I,i}$ is ergodic, the time average s_i is equal to the probability, in equilibrium, that an arriving task is rejected, i.e., an arriving task finds the process in state i+L (full queue). Stating this more precisely, if we let $X = X_{I,i}$ and $K = i + L$ then

$$s_i = \lim_{t\to\infty} \text{Prob}[X_t = K | \text{task arrives at time } t].$$

Due to the purely random nature of Poisson arrivals, it can be shown further (see [16], pp. 117-119) that the above coincides with the (unconditional) equilibrium probability p_K of X being in state K, that is,

$$s_i = p_K = \lim_{t\to\infty} \text{Prob}[X_t = K]. \tag{35}$$

Substituting back in (34), we have the pleasant (and somewhat intuitive) conclusion that

$$r_i = 1 - p_K = 1 - p_{i+L}\ . \tag{36}$$

In other words, the normalized average throughput rate of S_i (in equilibrium) is just the equilibrium probability of the queue not being full.

As S_i is an M/M/i/K queue, the general solution of p_K (35) is known (see [17], 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 (see Table 1). Moreover, the dependence on α and μ is only through

their ratio

$$u = \frac{\alpha}{\mu} \tag{37}$$

(the so-called "traffic intensity"). By (36), these remarks obviously apply as well to the solution of r_i which, in a general form, can be expressed as follows:

$$r_i = \frac{\left[1 - \left(\frac{u}{i}\right)\right] \sum_{n=0}^{i} \frac{u^n}{n!} + \frac{u^i}{i!}\left(\frac{u}{i}\right) - \frac{i^i}{i!}\left(\frac{u}{i}\right)^{L+i}}{\left[1 - \left(\frac{u}{i}\right)\right] \sum_{n=0}^{i} \frac{u^n}{n!} + \frac{u^i}{i!}\left(\frac{u}{i}\right) - \frac{i^i}{i!}\left(\frac{u}{i}\right)^{L+i+1}} \tag{38}$$

More compact and more meaningful expressions are obtained for specific instances of i, e.g., for i=1,2 we have the following solutions:

$$r_1 = \begin{cases} \frac{1 - u^{L+1}}{1 - u^{L+2}} & \text{if } u \neq 1 \\ \frac{L+1}{L+2} & \text{if } u = 1 \end{cases} \tag{39}$$

$$r_2 = \begin{cases} \frac{1 + \frac{u}{2} - 2\left(\frac{u}{2}\right)^{L+2}}{1 + \frac{u}{2} - 2\left(\frac{u}{2}\right)^{L+3}} & \text{if } u \neq 2 \\ \frac{2L + 3}{2L + 5} & \text{if } u = 2 \end{cases} \tag{40}$$

Generally, it can be shown 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). Accordingly, the limiting form of r_i, as $L \to \infty$, provides an upper bound (as a function u) on the value of r_i. Taking

formal limits of (38) for various restrictions on the value of $\frac{u}{I}$, we have

$$r_i = \begin{cases} 1 & \text{if } \frac{u}{I} \leq 1 \\ \frac{i}{u} & \text{if } \frac{u}{I} > 1 \end{cases} . \tag{41}$$

Thus, for a very large buffer, the normalized average throughput rate in structure state i is determined solely by the value of $\frac{u}{I} = \frac{\alpha}{i\mu}$ (the "utilization factor"; see [16], for example). If the task arrival rate α does not exceed the capacity $i\mu$, then almost all tasks are processed; if $\alpha > i\mu$ then r_i is approximately equal to the "normalized capacity" $\frac{i}{u} = \frac{i\mu}{\alpha}$. Although we could examine the functional properties of r_i in greater detail, they are relatively 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 (38) assume constant values for fixed values of the base model parameters, by (31) the performance variable Y can be expressed as a linear combination of lower level random variables, viz.

$$Y = \frac{1}{t} \sum_{i=1}^{N} r_i \, W_t^i \tag{42}$$

where W_t^i (25) 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 (42) is extended to include state i = 0 where, trivially, $r_0 = 0$ (see (30)), 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 [18]) 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 (21).

To clarify the approach we adopt in solving F_Y, it is helpful to rephrase equation (42) in terms of a model hierarchy for X_R (see [12], Def. 3). More specifically, if the sequence of variables $(W_t^1, W_t^2, \ldots, W_t^N)$ is identified with the level-0 model of a hierarchy (level-0 is closest to the "top") then (42) can be restated as the level-0 based capability function γ_0, i.e.,

$$\gamma_0(w_1, w_2, \ldots, w_N) = \frac{1}{t} \sum_{i=1}^{N} r_i w_i. \tag{43}$$

where w_i is the value of variable W_t^i ($w_i \in [0,t]$). If, further, we let B_y denote the set of accomplishment levels accounted for directly by the PDF F_Y, i.e.,

$$B_y = \{a \mid a \leq y\}$$

then

$$\begin{aligned} F_Y(y) &= \text{Prob}[Y \in B_y] \\ &= \text{Prob}[\gamma_0(W_t^1, W_t^2, \ldots, W_t^N) \in B_y]. \\ &= \text{Prob}[(W_t^1, W_t^2, \ldots, W_t^N) \in \gamma_0^{-1}(B_y)]. \end{aligned} \tag{44}$$

In other words, if we could characterize the probabilistic nature of the level-0 model, the desired solution could be obtained by formulating the probability of the inverse image $\gamma_0^{-1}(B_y)$.

At this level, however, we find that a probabilistic characterization is difficult to obtain since the random variables $W_t^1, W_t^2, \ldots, W_t^N$ 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 (17)), 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 capability function, has a simpler probabilistic description. For the hierarchy 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 level-1 model to be the sequence of variables $(V_1, V_2, \ldots, V_N)$ where

$$V_i = \lim_{t\to\infty} W_t^i = \text{time spent in state } i \text{ during } [0,\infty). \tag{45}$$

Although this level-1 model is no less "abstract" than the level-0 model, it should be clear that it contains more information, thereby admitting a well-defined "interlevel translation" (see [12]) from level-1 to level-0. When this translation is composed with γ_0, the resulting capability function γ_1 (based on the level-1 model) can be formulated as follows. Let v_i denote the value of V_i ($v_i \in [0,\infty)$) and, for notational convenience, let σ_j denote the sum

$$\sigma_j = \sum_{i=j}^{N} v_i \quad ; \quad 1 \leq j \leq N. \tag{46}$$

Then it is relatively easy to verify that

$$y_1(v_1,v_2,\ldots,v_N) = \begin{cases} \frac{1}{t}\sum_{i=1}^{N} r_i v_i, & \text{if } \sigma_1 \le t \\ \frac{1}{t}\sum_{i=j+1}^{N}(r_i - r_j)v_i + r_j, & \text{if } \sigma_{j+1} \le t,\ \sigma_j > t \\ r_N, & \text{if } \sigma_N > t\ . \end{cases} \tag{47}$$

At the cost of a more complicated capability function (compare (47) with (43)), we are now at a level where a probabilistic characterization is easier to obtain. This, in turn, can provide the solution we seek, for arguing as we did at level-0 (see (44)), we have

$$F_Y(y) = \text{Prob}[(V_1,V_2,\ldots,V_N) \in y_1^{-1}(B_y)]. \tag{48}$$

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 i and terminating in the absorbing state 0. For each state $i > 0$ that is visited, the variable V_i (45) is thus the time of a single "sojourn" is state i. Moreover, since X_R is a Markov process, it is known (see [15], 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

$$U = \text{sequence of states (excluding 0) visited during } [0,\infty). \tag{49}$$

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 $\ell \leq k$ and

$$U = (k, k-1, \ldots, \ell). \tag{50}$$

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

$$U = \Lambda \quad \text{(the null sequence)}. \tag{51}$$

(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

$$F_{Y|U} = \text{conditional PDF of Y given U=u} \tag{52}$$

then, by a well known formula, $F_Y(y)$ may be expressed as

$$F_Y(y) = \sum_u F_{Y|U}(y|u)\,\text{Prob}[U=u] \tag{53}$$

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 $F_{Y|U}(y|u)$ and further simplifying notation, let the level-1 variables V_i (45) be denoted by the single vector-valued variable

$$V = (V_1, V_2, \ldots, V_N)$$

taking values $v = (v_1, v_2, \ldots, v_N)$, and let C_y denote the inverse image of B_y under the capability function γ_1 (47), i.e.,

$$C_y = \gamma_1^{-1}(B_y) = \{v | \gamma_1(v) \leq y\}. \tag{54}$$

Then, in view of (48), when Y is conditioned by the event U = u,

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

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, i.e., if we let

$f_{V|U}$ = conditional joint pdf of V given U=u (55)

then

$$F_{Y|U}(y|u) = \int_{C_y} \int \cdots \int f_{V|U}(v|u)\, dv_1\, dv_2 \ldots dv_N \,. \tag{56}$$

Regarding (56), 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 (16) 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

$$f_{V|U}(v|u) = \begin{cases} \prod_{i \in u} \lambda_i e^{-\lambda_i v_i} & \text{if } v_j = 0, \text{for all } j \notin u \\ 0 & \text{if } v_j > 0, \text{ for some } j \notin u. \end{cases} \tag{57}$$

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

$$f_{V|U}(v|\Lambda) = \begin{cases} 1 & \text{if } v_1 = v_2 = \ldots = v_N = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{58}$$

Performing the indicated integration, on the other hand, is generally quite difficult, due to the nonlinear form of the capability function y_1 (see (47)). (Results obtained for N = 2 will be illustrated momentarily.)

As for the second product term in equation (53), 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 X obtained by sampling X_R each time it changes state. Moreover, by inspection of X_R,

if $i \geq 2$, then $\text{Prob}[\overline{X}_{n+1} = i-1 | \overline{X}_n = i] = c_i$ (see (17)) and $\text{Prob}[\overline{X}_{n+1} = 0 | \overline{X}_n = i] = 1 - c_i$; if $i = 1$, $\text{Prob}[\overline{X}_{n+1} = 0 | \overline{X}_n = 1] = 1$. Accordingly, if we let $\{p_i | 0 \leq i \leq N\}$ denote the initial state probability distribution of X_R, i.e.,

$$p_i = \text{Prob}[X_{R,0} = i] \tag{59}$$

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

$$\text{Prob}[U = u] = \begin{cases} p_k c_k c_{k-1} \cdots c_{\ell+1}(1 - c_\ell) & \text{if } k > \ell \geq 2 \\ p_k c_k c_{k-1} \cdots c_2 & \text{if } k > \ell = 1 \\ p_k(1 - c_k) & \text{if } k = \ell \geq 2 \\ p_1 & \text{if } k = \ell = 1 . \end{cases} \tag{60}$$

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

$$\text{Prob}[U = \Lambda] = p_0. \tag{61}$$

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

1) For each structure state i, apply (38) to determine the equilibrium solution of the normalized average throughput rate in state i.
2) For each state sequence u, apply (57), (58) to determine the conditional joint pdf of V given U = u.
3) For each pdf obtained in 2), apply (56) to determine the PDF of Y given U = u.
4) For each possible state sequence u, apply (60), (61) to determine the probability that U = u.
5) Combining the results of 3) and 4), apply (53) to determine the PDF F_Y for 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).

		SYMBOL	NAME	DEFINITION
ENVIRON-MENT		α	task arrival rate	(8)
COMPUTER	BASIC	N	number of processors	(9)
		L	buffer capacity	(10)
		λ_p	processor failure rate	(12)
		c_p	processor coverage	(13)
		μ	processor processing rate	(18)
		λ_b	buffer stage failure rate	(14)
	DERIVED	λ_B	buffer failure rate	(15)
		λ_i	transition rate from structure state i	(16)
		c_i	coverage in structure state i	(17)

Table 1. Base model parameters.

PDF of performance variable Y: $F_Y(y) = \text{Prob}[Y \leq y]$

1) $0 \leq y < r_1$

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

2) $r_1 \leq y < r_2$

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

3) $r_2 \leq y \leq 1$

$$F_Y(y) = 1$$

KEY

t = duration of utilization

$\lambda_2 = 2\lambda_P + L\lambda_b$

$\lambda_1 = \lambda_P + L\lambda_b$

$c_2 = \frac{2\lambda_P c_P}{\lambda_2}$

$\nu_i = \frac{\lambda_i t}{r_i}$

p_i = initial state probability, structure state i

r_i = long run average throughput rate, structure state i

Table 2. Closed-form solution of F_Y.

Step 1)

The equilibrium solutions r_1 and r_2 have already been considered and are given by equations (39) and (40).

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 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. Letting f_i denote the pdf of V given $U = u_i$ and applying (57), (58) we have

$$f_1(v|u_1) = \lambda_1 e^{-\lambda_1 v_1} \lambda_2 e^{-\lambda_2 v_2} ,$$

$$f_2(v|u_2) = \begin{cases} \lambda_2 e^{-\lambda_2 v_2} & \text{if } v_1 = 0,\ v_2 \geq 0 \\ 0 \text{ if } v_1 > 0 , \end{cases}$$

$$f_3(v|u_3) = \begin{cases} \lambda_1 e^{-\lambda_1 v_1} & \text{if } v_1 \geq 0,\ v_2 = 0 \\ 0 \text{ if } v_2 > 0 , \end{cases}$$

$$f_4(v|u_4) = \begin{cases} 1 \text{ if } v_1 = v_2 = 0, \\ 0 \text{ otherwise.} \end{cases}$$

Step 3)

To obtain the integrals (56) of these pdf's over the region $C_y = \{v|Y_1(v) \leq y\}$, 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_1 (47) 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 solutions, let F_i denote the PDF of Y given $U = u_i$ (52) and let ν_i denote the quantity

$$\nu_i = \frac{\lambda_i t}{r_i} \tag{62}$$

which, when r_i is fully expressed, is a function of base model parameters L, α, and μ as well as λ_i and t. Then, for the instance where i = 1 and y is in the range $r_1 \leq y < r_2$, the integration according to (56) of f_i over C_y (see Fig. 5) yields the solution

$$F_1(y) = 1 - \left[e^{-\nu_2 y} + \frac{\nu_2 \left(e^{-\nu_1 y} \, e^{\frac{-(\nu_2-\nu_1) r_2 (y-r_1)}{(r_2-r_1)}} - e^{-\nu_2 y} \right)}{(\nu_2-\nu_1)} \right].$$

Solutions of $F_1(y)$ in other ranges of y and solutions of the other F_i are obtained in a like manner.

Step 4)

By the definitions of u_1 - u_4 and on applying (60), (61), 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 5)

Applying equation (53) to the results obtained in steps 3) and 4), a closed-form solution of F_Y, expressed in terms of y, r_i and ν_i (see (62)), is displayed in Table 2. For the reader interested in seeing the solutions that were obtained as a result of Step 3), we note that they may may be resurrected by considering the following special cases of F_Y:

$$F_Y(y) = \begin{cases} F_1(y) \text{ if } p_2 = 1,\ c_2 = 1 \\ F_2(y) \text{ if } p_2 = 1,\ c_2 = 0 \\ F_3(y) \text{ if } p_1 = 1 \\ F_4(y) \text{ if } p_0 = 1\ , \end{cases}$$

Given this solution of F_Y, we have thus obtained a closed-form solution of the performability p_S for intervals of the form $B_y = \{a | a \leq y\}$, 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 $t = 10$, $u = \frac{\sigma}{\mu} = 1.5$, $\lambda_p = 0.01$, $\lambda_b = 0.001$, $c_p = 0.99$, $p_2 = 0.9$, $p_1 = 0.09$, and $p_0 = 0.01$; the figure furnishes several plots showing how $F_Y(y)$ varies as L ranges from 1 to 25 in steps of 2. Fig. 7 is similar to Fig. 6 except that $p_2 = 1.0$ while $p_1 = p_0 = 0.0$.

Finally, as an illustration of how such a closed-form solution can be used to examine design tradeoffs, the buffer capacity L is an example of a design choice which influences both 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 (40)). On the other hand, if reliability is 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 $B = \{y|y > 0.8\}$, i.e., the system S performs within B if the normalized average throughput rate is greater than 0.8. Then, for the parameter values of Fig. 6, the variation of $p_S(B)$ as a function of buffer capacity L is displayed in Fig. 8. In particular, we see that the optimum buffer capacity is 5 for this choice of parameter values.

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, Queuing 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-22, pp. 501-509, June 1980.

[12] J. F. Meyer, "On evaluating the performability of degradable computing systems," IEEE Trans. Comput., vol. C-22, 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, Jan. 1981.

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

[16] L. Kleinrock, Queuing Systems, Volume I: Theory. New York: John Wiley & Sons, 1975.

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

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

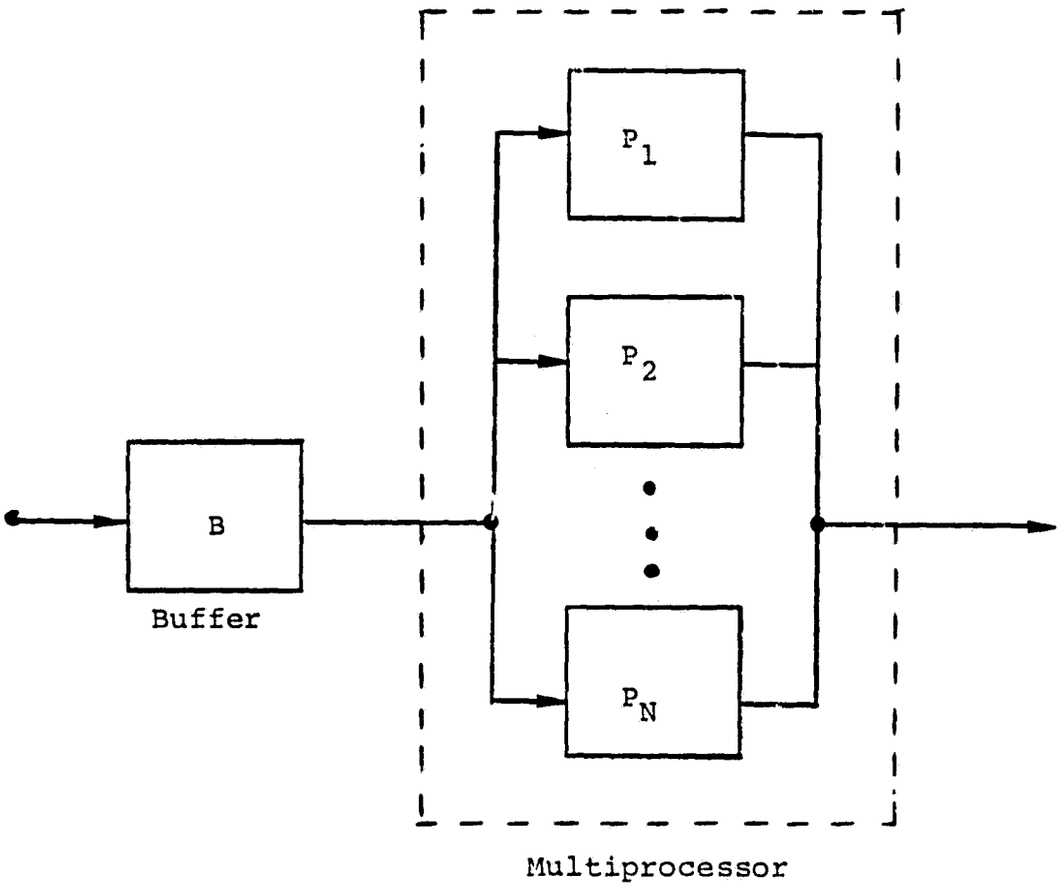

Fig.1. Block diagram of C.

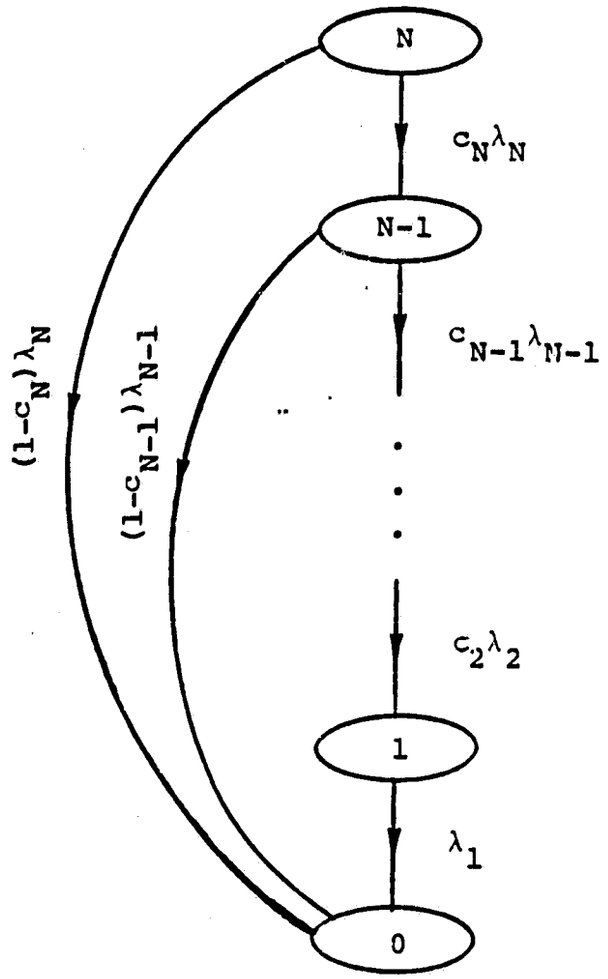

Fig.2. State-transition-rate diagram of X_R.

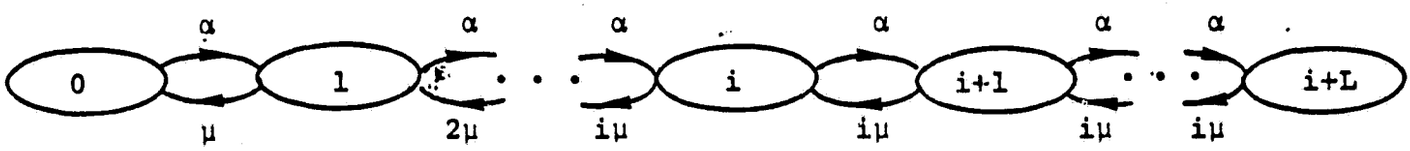

Fig.3. State-transition-rate diagram of $X_{I,i}$.

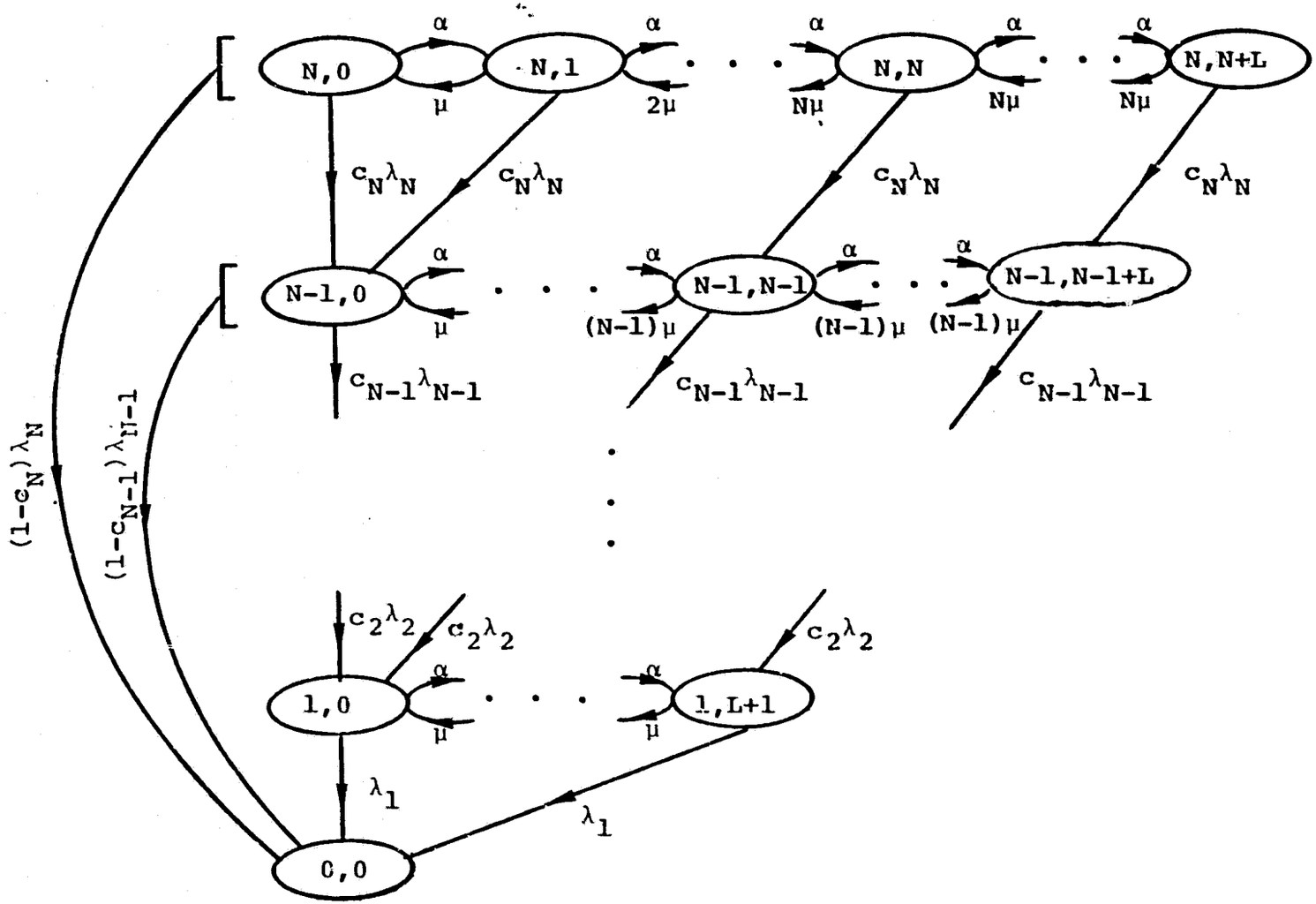

Fig.4. State-transition-rate diagram of X_C.

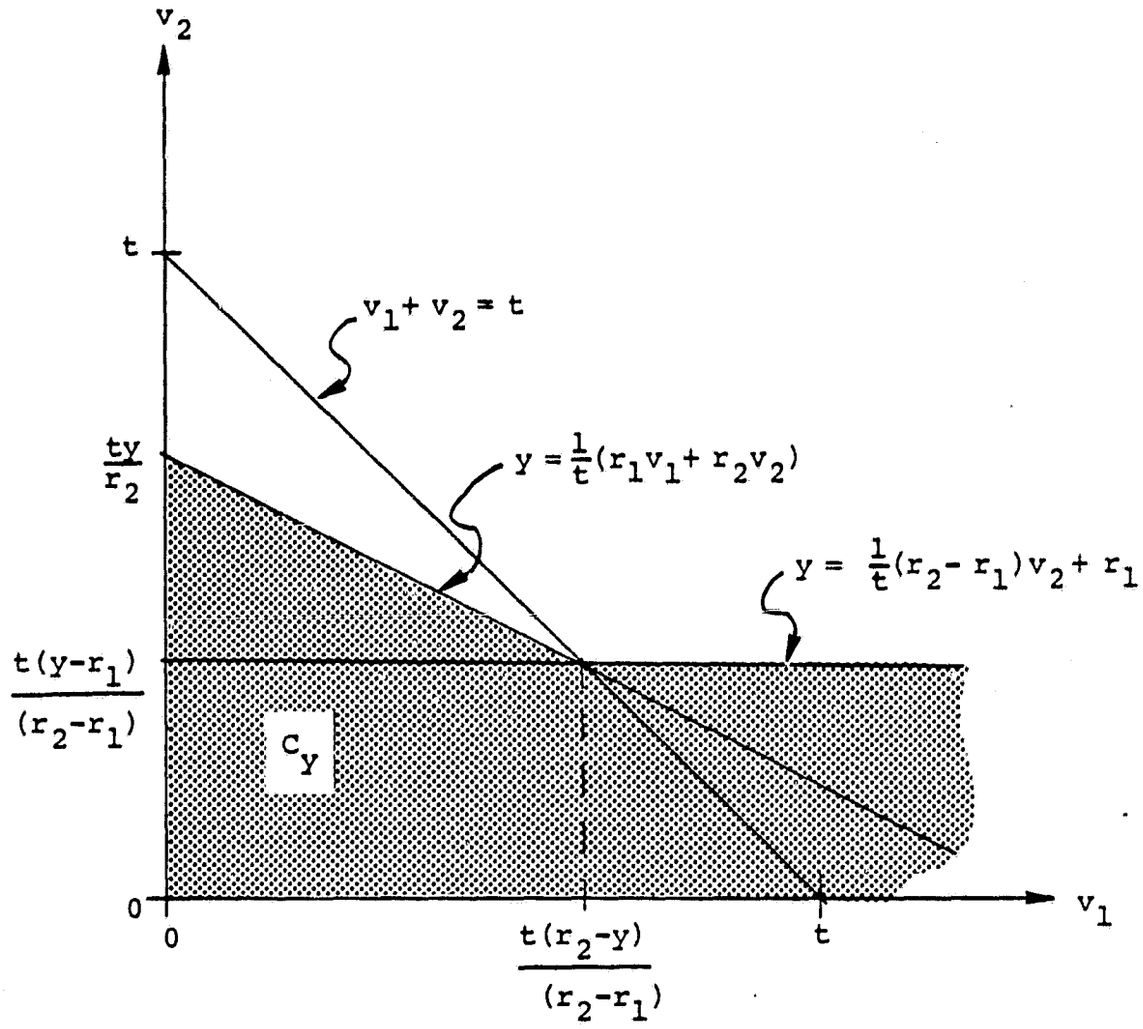

Fig.5 C_y for y in the range $r_1 \leq y < r_2$.

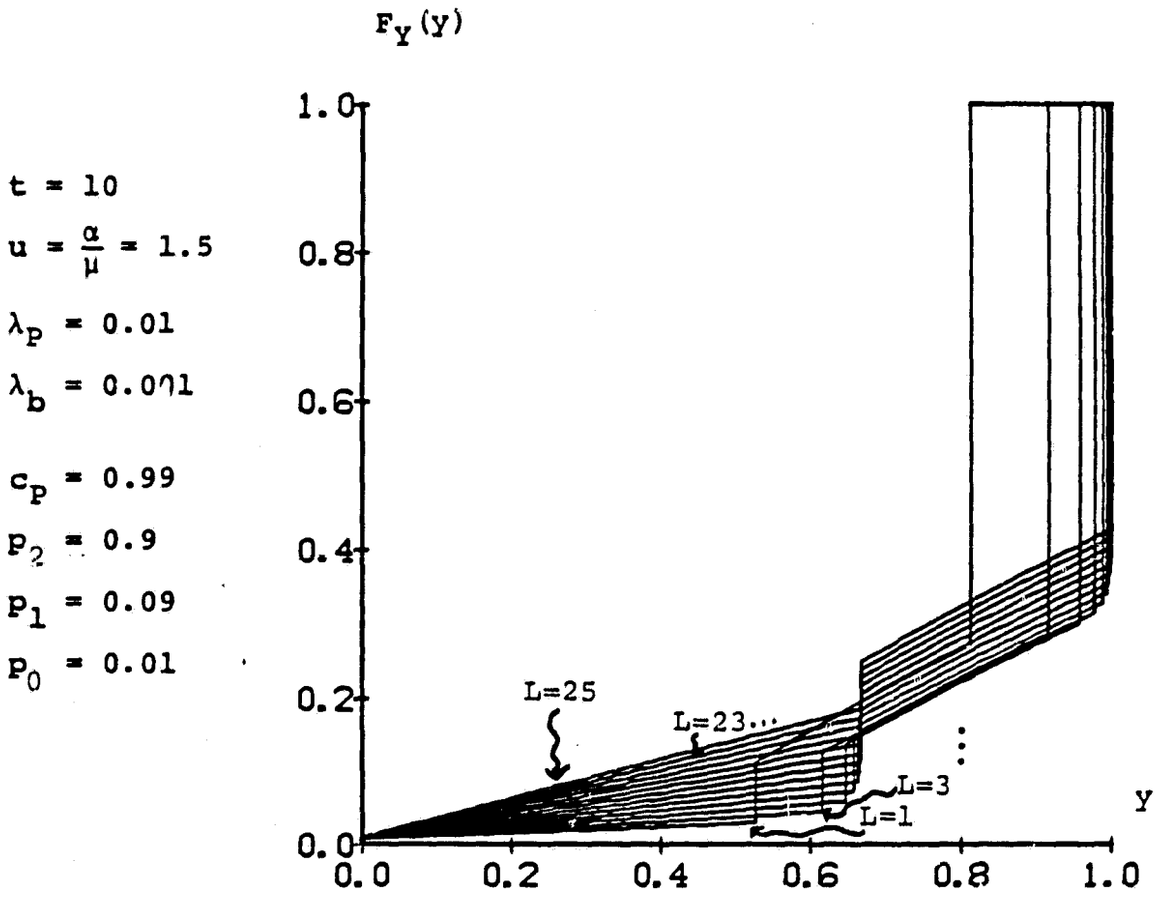

Fig. 6. Plot of $p_S(B_y)=F_Y(y)$ as a function of y for the choices of t and the base model parameters shown above. L varies from 1 to 25 in steps of 2.

$t = 10$

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

$\lambda_p = 0.01$

$\lambda_b = 0.001$

$c_p = 0.99$

$p_2 = 1.00$

$p_1 = 0.00$

$p_0 = 0.00$

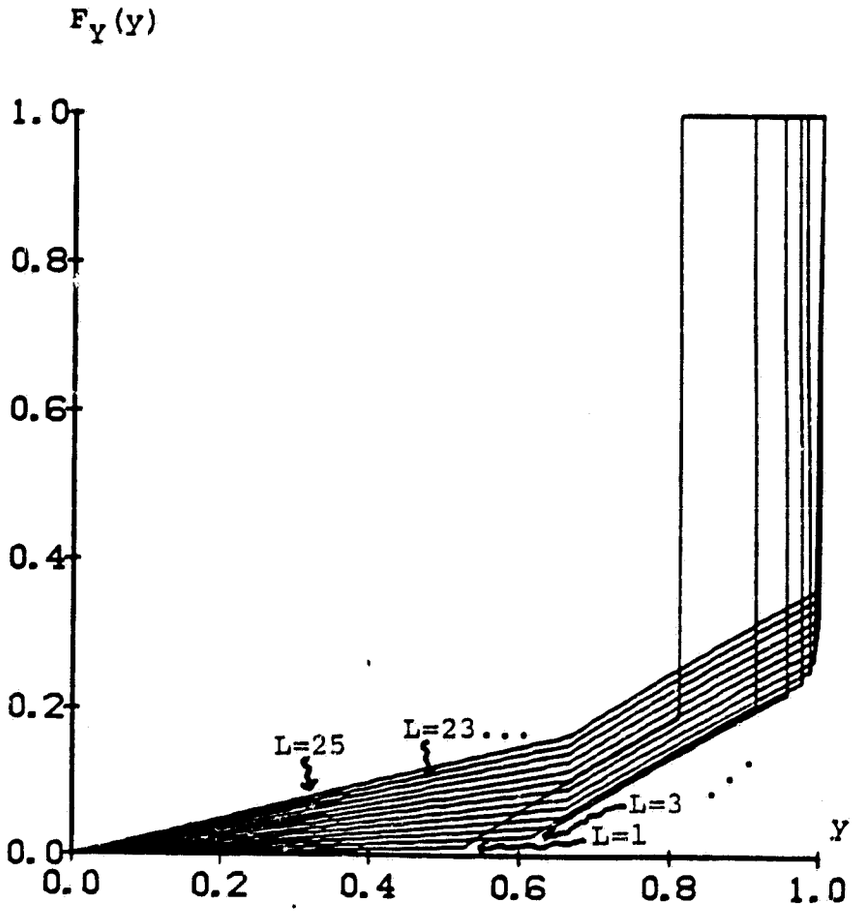

Fig. 7. Plot of $p_S(B_y)=F_Y(y)$ as a function of y for the choices of t and the base model parameters shown above. L varies from 1 to 25 in steps of 2.

$p_S(B)$ as a function of the buffer capacity L, where $B=\{ a \mid a > .8\}$.

$t = 10$

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

$\lambda_p = 0.01$

$\lambda_b = 0.001$

$c_p = 0.99$

$p_2 = 0.9$

$p_1 = 0.09$

$p_0 = 0.01$

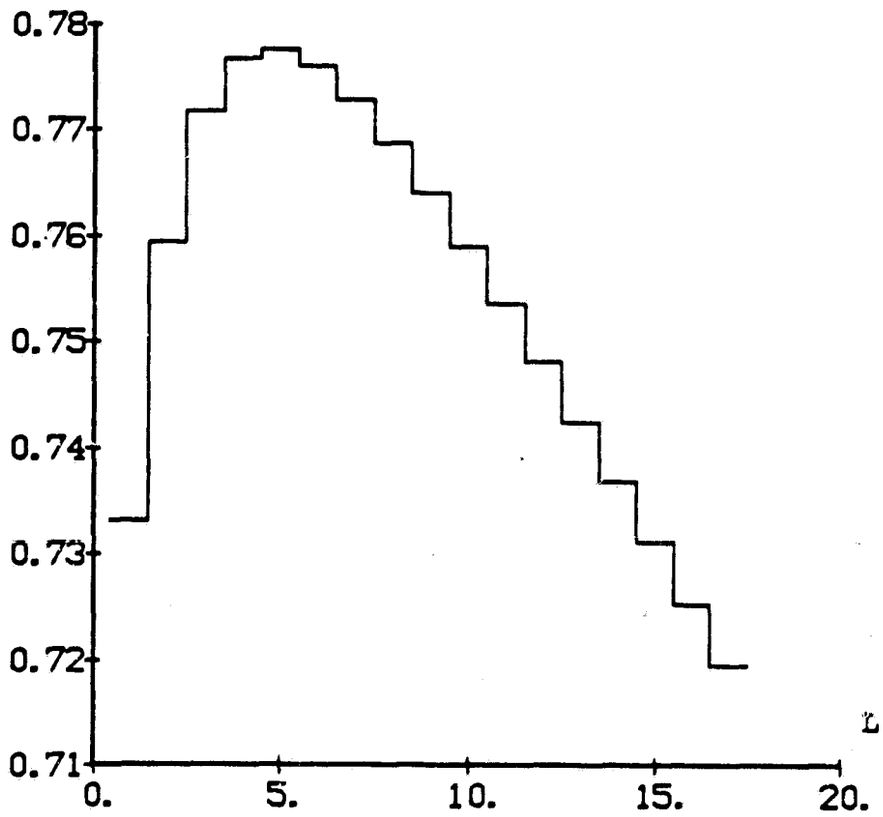

Fig. 8. $p_S(B)$ as a function of buffer capacity L.