Multiprocessor Performability Analysis

Noe Lopez-Benitez, Member IEEE
Texas Tech University, Lubbock
Kishor S. Trivedi, Fellow IEEE
Duke University, Durham

Key Words — Fault tolerant system, Performability, Processor array, Markov model, Stochastic Petri net.

Reader Aids — General purpose: Present modeling techniques
Special math needed for explanations: Probability, Markov models
Special math needed to use results: Same
Results useful to: Reliability analysts and multiprocessor designers

Summary & Conclusions — Performability models of multiprocessor systems and their evaluation are presented. Two cases in which hierarchical modeling is applied are examined.

1. Models are developed to analyze the behavior of processor arrays of various sizes in the presence of permanent, transient, intermittent, and near-coincident faults. Models can be generated for typical reconfiguration schemes that consider the failures of several types of components (detailed modeling). These models consider a survivability factor derived in terms of the physical distribution of faulty components. Capacity-based reward rates are then used to derive overall performability measures.

2. Queueing network models are solved to derive performance measures that are used as reward rates within an overall Markov failure-repair model of bus-based multiprocessor systems. Several configurations are compared in terms of their performability. In both cases, Markov models are generated using MGRE and solved using SHARPE. The analysis, particularly for case 2, is by no means exhaustive as several parameters are involved in the overall model. However, the hierarchical models shown, combined with the use of diverse tools such as MGRE & SHARPE, facilitate the analysis of large systems in various environments. The models can be fine-tuned according to specific applications and performance measures.

1. INTRODUCTION

Performability models are used to combine a reliability model and a performance model. A reliability model represents the behavior of the system in response to faults. For a system of several components that can fail, an operational-state or a structure-state can be represented by the set of remaining operational components. The performance model determines levels of performance [3] (reward rates) when the system is operational. Throughout its operational life the system can reconfigure (in response to failures) to a new configuration which has a degraded performance; such degraded levels of performance depend on the computational resources available in the new configuration. By assigning a reward rate to each structure-state of the reliability model, the effect of failures on the performance of the system can be predicted.

Acronyms¹ & Nomenclature

MGRE Model Generator and Reliability Evaluator
SHARPE Symbolic Hierarchical Automated Reliability & Performance Evaluator
MRM Markov reward model
MSPN modified stochastic Petri nets
FTPA fault-tolerant processor arrays
SRE successive row elimination
ARCE alternate row-column elimination
PE, IOL, S, BL, HL, SBL processing element, I/O link, switch, bypass link, horizontal link, spare bypass link.

Notation

\[ M_q \]
\[ M_0 \]
\[ m_i \]
\[ M \]
\[ t_i \]
\[ T \]
\[ Pr_{t_i} \]
\[ Pr \]
\[ S_t(M_q) \]
\[ M^* \]
\[ c_i \]
\[ b_{ij} \]
\[ B_i \]
\[ B \]
\[ N \]
\[ \rho \]
\[ \alpha_i \]
\[ \nu_{ij} \]
\[ r_{ij} \]
\[ \delta_{ij} \]
\[ \gamma_{ij} \]
\[ \mu_{ij} \]
\[ \Pr \]
\[ P, M, D \]
\[ (n_0, n_1, n_2) \]

¹ The singular & plural of an acronym are always spelled the same.
by a Markov model and a reward rate is attached to each state. The failure/repair/reconfiguration behavior is captured of the Markov chain. The reward rates can be:

\[ E\{X(t)\} \]

The mean accumulated reward in 

\[ \pi_i \text{ steady-state probability of state } i \]

\[ P_i(t) \text{ Pr}\{\text{Markov chain is in state } i \text{ at time } t\} \]

\[ A_C(t) \text{ computational availability} \]

\[ 3(\cdot) \text{ indicator function: } 3(\text{True})=1, 3(\text{False})=0. \]

The most common type of performability model is MRM where the failure/repair/reconfiguration behavior is captured by a Markov model and a reward rate is attached to each state of the Markov chain. The reward rates can be:

- binary as in a reliability/availability model
- based on resource capacity as in [4]
- based on system performance index in the state [5,6].

Once state probabilities of the Markov chain are computed we can derive important performability measures. The mean reward rate is:

\[ E\{X(t)\} = \sum_i r_i P_i(t), \]

\[ E\{X(\infty)\} = \sum_i r_i \pi_i. \]

The mean accumulated reward in \([0,t]\) is:

\[ E\{Y(t)\} = \sum_i r_i \int_0^t P_i(\tau) d\tau. \]

Since \(X(t)\) is a discrete r.v., its Cdf is:

\[ \text{Pr}\{X(t) \leq r\} = \sum_i S_F r P_i(t). \]

On the other hand, computation of the Cdf of the continuous r.v. \(Y(t)\) is quite involved [6]. For the special case of \(Y(\infty)\), Beaudy [4] provided a simple method which has been recently generalized in [7].

With binary reward assignment,

- \(E\{X(\infty)\}\) reduces to steady-state (un)availability,
- \(E\{X(\tau)\}\) specializes to instantaneous (un)availability,
- \(Y(t)\) becomes the uptime (downtime) in \([0,t]\),
- \(Y(t)/t\) becomes interval (un)availability,
- \(E\{Y(\infty)\}\) represents system mean time to failure
- \(\text{Pr}\{Y(\infty) > t\}\) is system reliability at time \(t\).

An important performability measure proposed in [4] is \(A_C(t)\). If \(r_i\) is set to the system capacity in state \(i\), then \(E\{X(\tau)\} = A_C(t)\). Subsequently, many authors have proposed reward assignments based on a system performance model and computed \(E\{X(\tau)\}, E\{Y(\tau)\}, \text{Cdf}\{\tau\}, \text{Cdf}\{\infty}\) [8].

The main questions that we need to address in the practical application of the above methods is how to: 1) generate & solve a large Markov model, and 2) compute & assign the reward rates to the states of the Markov model. If the Markov model is medium-sized, it can be hand generated and solved by one of the many tools available. One such tool is SHARPE; we use this tool for the computations in this paper. If the Markov model is rather large, hand generation is error-prone or infeasible; we can use a concise stochastic Petri net to specify the model and use a tool to generate automatically the underlying Markov chain. We can use: 1) a general purpose stochastic Petri net solver such as GreatSPN [9], SPNP [10], or SHARPE, or 2) a special purpose SPN tool tailored to model the behavior of fault-tolerant processor arrays. One such tool is MGRE [1] which is used in this paper.

The use of SPN does not avoid the generation & solution of a large Markov state space; it merely shields the user from the largeness problem. Alternatively, we are often able to decompose the problem hierarchically and avoid the largeness problem [2]. Hierarchical techniques allow the decom position of large models into several levels; at each level, reduced models are solved independently and their solutions are aggregated at a higher level. Applications of these techniques include CARE III & HARPE [11]. This paper shows the use of such a hierarchical approach in modeling different failure modes.

The remaining question is the computation of reward rates. The simplest choice is to use the 'number of up resources in a state' as a reward assignment. We use such capacity-based reward assignment to analyze two well-known processor arrays: SRE & ARCE [12]. First, performability is analyzed under the assumption that only processing elements fail. Using HARPE, the behavior of processor arrays of various sizes are analyzed in the presence of permanent, transient, and intermittent faults with the underlying assumption that near-coincident faults can occur. Next, the failure of component-types in the array other than processing elements, (eg, switches, links) is considered (detailed modeling). MGRE is used to generate the large state-space involved in detailed models; these models are then solved using SHARPE.

A more detailed performability study uses a performance model. For each structure state, an appropriate performance measure is computed and then used as a reward assignment. This is another use of hierarchy in which the lower.level model is a performance model and the top-level model is a failure/repair model. Several bus-based multiprocessor configurations are analyzed where the performance model is a product-form queueing network [13]. Large failure/repair Markov models are obtained using MGRE and then solved using HARPE.

Section 2 —

- discusses MSPN-based fault models of processor arrays,
- discusses a fault handling model to account for near coincident, permanent, transient, and intermittent faults,
- reports performability results based on capacity-based reward assignments
Section 3 —

- introduces a simple MSPN-based model for multiprocessor systems,
- discusses a closed queueing network to derive performance rewards,
- compares several multiprocessor configurations.

2. PROCESSOR ARRAYS

The SRE & ARCE reconfiguration schemes are compared. Since MGRE is used to generate the Markov chains used in SHARPE, the MSPN which describes the fault behavior of SRE, is described briefly; a detailed description of MSPN derived for SRE & ARCE is in [12]. Ref [1] describes an MSPN as extending an SPN [15] in order to represent FTPA.

Notation (For the MSPN definition only)

\[ P = \{ p_1, \ldots, p_n \} \] set of places
\[ A = \{ T \times P \} \cup \{ P \times T \} \] set of input & output arcs
\[ S_i = \{ S_i \} \] set of sequences \( S_i \) of transitions that fire immediately after an exponential firing of transition \( t_i \in T \).

MSPN = \( (P, T, A, M_0, Pr, Sq, B, Cv) \)

Specifically, an MSPN accounts for probabilities of different fault types by associating with each \( t_i \), the set of attributes:

\[ (Pr_{t_i}, S_i(M_q), B_i, c_i) \]

If no immediate firing is required then \( S_i(M_q) \) is null. Depending on the reconfiguration scheme, \( S_i \) can be unique for all markings or can be determined in terms of \( M_q \).

2.1 Fault Models

Figure 1 depicts an MSPN of the SRE reconfiguration scheme which describes the fault behavior of the array whose schematic layout is shown in figure 2. The SRE array initially functions with \( n \times n \) PE. If a single PE is detected as faulty, SRE eliminates the row with the faulty PE. Thus, the performance of the array degrades gracefully when faulty PE are detected and the array reconfigures successfully. The minimum working configuration corresponds to 1 fault-free row. The MSPN describes this behavior by letting the firing of transition \( t_1 \) represent the exponentially distributed occurrence of PE faults. Furthermore, if other components are allowed to fail, \( B_i \) identifies those places that contain components whose failure causes \( t_i \) to fire: \( b_1 = b_2 = b_3 = 1 \) indicate that the failure of a PE; an IOL, or an HL causes \( t_i \) to fire. The number of tokens in the input places are modified according to the current values of their variable multiplicities [1] also shown in figure 1. The failure of a switch or the failure of an active BL is fatal and is represented by the firing of \( t_2 \). In this case the number of tokens in all places are reset to 0, as indicated by the arc multiplicities labeled \( all \). The firing of \( t_3 \) models the failures of SBL.

A similar procedure is used to obtain a model for ARCE. In this scheme, not just rows are eliminated but also columns in an alternate fashion which leads to a minimal working configuration consisting of 1 processing element. Details on this reconfiguration scheme and its fault model, are in [12] & [14], respectively.

Figure 1. MSPN of SRE

Figure 2. Schematic Layout of SRE

2.2 Probabilities of Survival

If SBL are not fault-free, any operational configuration (described by a \( M_q \)) contains \( N = \#\text{PE}_q \) faulty SBL that can hinder a successful reconfiguration when the faulty row to be eliminated contains at least 1 faulty SBL. This is represented in the MSPN of figure 1 via the attributes associated with transitions. \( Pr_{t_1}^{\text{SBL}} \) is associated with transitions to new operational marking which occur when \( t_1 \) fires.
The probability $Pr_{f|t_i}$ is associated with transitions to the failure marking which in this case are generated by the firing of $t_2 \in S_1$. Let $\rho = Pr_{f|t_i}$; then $Pr_{1-f|t_i} = 1 - \rho$ [16]. In general, $\rho$ corresponds to the sum of all $Pr_{f|t_i}$ for all $t_i$ that generate the same operational marking.

An estimate of the survivability factor can be obtained using the hypergeometric pmf:

$$Pr\{S=s\} = \binom{y}{s} \cdot \binom{z}{N-s} / \binom{N}{y}.$$ (1)

$y = \text{number of remaining active rows}$

$z = \text{number of remaining active columns}$

$S = \text{a random number of faulty SBL in the row to be eliminated.}$

Eq (1) calculates $Pr\{\text{S faulty SBL exist in the row to be eliminated}\}$. The total number of SBL in a current configuration is $y-z$. Since, in SRE, reconfiguration succeeds only if no faulty SBL exist in the row to be eliminated, the probability of survival is:

$$\rho = Pr\{S=0\} = \binom{y-z}{0} / \binom{y}{y-z}.$$ (2)

2.3 Fault Handling Model

Another attribute associated with transitions in the MSPN, has the purpose of aggregating $c$ to the underlying Markov chain. The user must specify these factors for each $t_i$. For SRE, we let $c=1$ for $t_1$ because we assume that the failure to detect faulty SBL does not cause the array to fail. However, such failures are considered at the time the array attempts to reconfigure because $t_1$ fires. For $t_2$ we can as well let $c=1$, because the resulting marking is a failure marking anyway. So, we have to calculate the coverage factor for $t_1$ which in this case is specified as a function $c(x)$ where $x$ is a function of the current marking and the set of failure rates selected.

$$x = \sum_{i=1}^{k} (m_i - 1) \cdot a_i \cdot b_i,$$ (3)

$m_i-1 = \text{number of remaining components whose failure during reconfiguration gives rise to a near-coincident fault condition.}$

If the occurrence of a near-coincident fault causes the array to fail, then $x$ is the failure rate of the array due to near-coincident faults; $b_i \in B_i$ simply selects those components involved in the firing of $t_i$.

A general fault-handling model used to determine $c(x)$ is described in figure 3. This model extends the one proposed in [17].

**Notation**

- $\text{det}$: detection state.
- $\text{ptr}$: fraction of detected faults that are transient.
- $\text{pitr}$: fraction of detected faults that reach state $\text{intmt}$; they are intermittent type faults.

**Figure 3.** Markov Model of the Fault Handling Process

**Assumptions**

1a. Detection time is exponentially distributed; its mean $1/\delta = 5$ msec.

1b. Transient faults return to the $\text{up}$ state which represents the current operational configuration.

2. Intermittent faults go to the $\text{intmt}$ state. The time in this state is exponentially distributed; its mean $1/\gamma = 10$ msec. The array recovers with probability $r$, or the intermittent faults become permanent with probability $1-r$.

3. Reconfiguration time is exponentially distributed; its mean $1/\mu_r = 1$ sec. The array reaches the $\text{degr}$ state with probability $q = 0.99$, or it fails with probability $1-q$.

4. In order to consider the effect of near-coincident faults, a transition is added from states $\text{degr} & \text{intmt}$ to state $\text{F}$, each with a rate (3). The transition rate from state $\text{rec}$ to state $\text{F}$ is incremented by $x$. The $c(x)$ is computed as $Pr\{\text{the Markov chain in figure 3 eventually enters state } \text{degr}\}$. The conditional probability that the array fails due to the occurrence of a fault is determined by SHARPE using:

$$f(x) = 1 - c(x) - u(x).$$

2.4 Reward Models

**Notation** (Local)

- $k$: number of active processors
- $\text{rwd}(k)$: reward functions.

Reward models are automatically generated by MGRE. For processor arrays we use as reward rates the computational capacity (the number of PE active in an operational configuration). Thus, MGRE generates the MRM, viz., the reliability model plus a sequence of reward functions. These functions are then specified in a SHARPE input file as $\text{rwd}(k) = k$; $\text{rwd}(k)$ can be assigned values based on any other appropriate performance measure. If $\text{rwd}(k) = 1$, then the model corresponds to a reliability model.

2.5 Performability Results

The performability models generated by MGRE and the fault handling model of figure 3 are solved in a hierarchical
manner using SHARPE to obtain the results shown in figures 4 - 10; the models use a PE failure rate, \( \alpha_{PE} = 10^{-4} \)/hour.

Figure 4 shows the mean reward rate at time \( t \) (Computational Availability) of SRE arrays when only the PE fail. Initially, the maximum reward rate is \( n^2 \). As the array size \( n \) increases, the performability declines sharply within a relatively short mission time due to more points of failure. Figure 5, shows the mean accumulated reward in \([0,t]\) for several array sizes.

Performability improves when a fraction of detected faults are transient or intermittent, because reconfiguration is triggered only when a detected fault has become permanent.

Figures 6 & 7 plot the mean reward rates and the mean accumulated reward for a 12\( \times \)12 SRE array. Besides the effect of near coincident faults implicit in the fault-handling model, these plots reflect the effect of transient & intermittent faults.

Figure 8 shows the effect on the mean reward rate of SRE for a 12\( \times \)12 SRE array, assuming that components other than PE have a failure rate = \( 10^{-5} \)/hour. MGRE generates a state space consisting of 948 states. The plots in figure 8 were obtained assuming that permanent, transient, and intermittent faults can occur; the results are compared with the results of SRE model without detail under the same circumstances. Figure 9 compares SRE with ARCE for several array sizes. Only the PE fail when permanent, transient, and intermittent fault types can occur.
Figure 8. Mean Reward-Rate Comparison
[Detailed modeling of all fault - types; 12 × 12 SRE array; \( \alpha_{PE} = 10^{-4} \) /hour, \( \alpha_{other} = 10^{-6} \) /hour]

Figure 9. Mean Reward-Rate Comparison of SRE & ARCE Arrays
[Permanent, transient, intermittent fault-types; \( \alpha_{PE} = 10^{-4} \) /hour]

3. MULTIPROCESSOR SYSTEMS

We address the case of single bus-based shared-memory multiprocessor systems. These systems are designed to allow for graceful performance degradation when faulty units are detected. Performance measures include response time and throughput. Again, two models are combined and solved in a hierarchical fashion using SHARPE. A queueing network model is used to derive the performance measures which are then used as reward rates in a Markov failure-repair model. Unlike processor arrays, faulty processing units in a multiprocessor system can be repaired. However, to carry out repairs, the system must be first shutdown; therefore we concentrate our interest on the system's ability to operate without repair interruption. The Markov models are generated using MGRE and evaluated using SHARPE.

3.1 Reliability Model

Consider a multiprocessor system of \( P \) processors which operate concurrently. The processors are connected to \( M \) memory modules through a single bus. Also connected to the bus are \( D \) I/O devices. This configuration corresponds to bus-based multiprocessor architectures such as the Sequent 2000/700 machine. Any of these resources can fail, causing the system to degrade gracefully or fail if the system reaches the state in which either: 1) all processors have failed, 2) all memories have failed, or 3) all I/O devices have failed. Each state in the Markov chain can be described by \( (n_0, n_1, n_2) \).

Assumptions

1. All items have constant failure rates. Processor failure rate is \( \alpha_0 \); memory-unit failure rate is \( \alpha_1 \); I/O device failure rate is \( \alpha_2 \).
2. The coverage \( c \) is constant; it accounts for the probability of success of the fault detection and switching mechanisms when faults occur. Thus, imperfect coverage implies that the system can fail.
3. The bus is failure free.

Transitions between operational states in the Markov chain occur when a component say, type \( j \) fails. These transition rates are calculated as \( n_j \cdot \alpha_j \cdot c \). Transition rates to the failure state are:

\[
\alpha_f = \sum_j \left( c \cdot \left( \alpha_j - 1 \right) + (1 - c) n_j \cdot \alpha_j \right)
\]

Thus, a failure state is reached if: 1) a fault is not detected, or 2) all resources of at least one type needed to support normal execution of a process have failed. The entire model consists of \( P \times M \times D \) operational states and can be generated using the stochastic Petri net model in figure 10.

Figure 10. MSPN Model of the Multiprocessor System
[\( \nu_j \) = variable multiplicity associated with an arc from \( p_i \) to \( t_f \)]
In general for each \( p_i \), a variable multiplicity is determined:

\[

v_{ij} = 1, \text{ if } i = j; \quad v_{ij} = m_i \cdot \mathbb{1}(m_i = 1), \text{ otherwise}
\]

The \( \mathbb{1}(m_i = 1) \) is evaluated each time a transition \( t_i \) fires; \( m_i = n_i \). For example, suppose a marking has been reached in which there is only 1 processor functioning \( (n_0 = 1) \), then when \( t_0 \) fires, the remaining places are flushed to generate a null marking which identifies the failure state.

### 3.2 Performance Model

Queueing networks have been used extensively to predict performance measures of computer systems. A multiple server queueing network [13] described in figure 11 is used to model the 1-bus shared-memory system.

**Assumptions**

1. Up to \( K \) terminals submit jobs to the queueing network. The mean thinking-time for terminals is \( 1/\lambda \).
2. Each processor has a service rate of \( \mu \).
3. Although all data traffic goes through the bus, job submission & completion do not require additional waiting time at the bus.
4. The bus is a single fcfs service center with service rate \( \mu_b \).
5. Since we are interested in the mean response time and the throughput of the system for all types of jobs, the memory modules and the I/O devices are modeled as multiple service centers. The service rate for I/O devices is \( \gamma \), and the request rate for memory modules is \( \eta \).
6. The service time distributions and service disciplines are such that the network is of the product-form [13].

Upon completion, each job leaves the system with probability \( p_0 \) or generates a request to access either the memory system or the I/O devices; this request is routed through the bus with probability \( 1 - p_0 \). The memory system is accessed with probability \( q_0 \) and the I/O devices with probability \( q_1 \).

### 3.3 Performability Results

An initial system with \( (P,M,D) = (4,2,2) \) is analyzed. In addition, 2 updates of the original system are considered: \( (P,M,D) = (8,4,4) \), and \( (P,M,D) = (16,8,8) \). These systems are compared in terms of their reliability and 2 performance-measures: 1) Average number of jobs completed, and 2) system response time.

#### Parameter Values — Reliability Model

- Processor failure rate: \( \alpha_0 = 32 \text{ Fit} \) (\( \approx 1 \text{ failure/year} \)) [18]
- Memory-module failure rate: \( \alpha_1 = 64 \text{ Fit} \) (\( \approx 2 \text{ failures/year} \))
- I/O-device failure rate: \( \alpha_2 = 129 \text{ Fit} \) (\( \approx 4 \text{ failures/year} \))
- \( c = 0.99 \)

#### Parameter Values — For Performance Model

Jobs are submitted via 100 terminals connected to the system
- Mean terminal thinking-time: \( 1/\lambda = 15 \text{ sec} \)
- Processor service rate: \( \mu = 500 \text{ bursts/sec} \)
- Bus service rate: \( \mu_b = 5000 \text{ requests/sec} \)
- Memory service rate: \( \eta = 250 \text{ requests/sec} \)
- I/O-device service rate: \( \gamma = 15 \text{ requests/sec} \)

**Branching Probabilities:** \( p_0 = 0.20, q_0 = 0.90, q_1 = 0.10 \)

The queueing network is specified to SHARPE using the \( pfpm \) model type. The reward assignments are throughput-based and response time-based. In the latter case, the average response time in a system failure state is infinite, and since our Markov reward models can not accommodate infinite reward rates, we use the reciprocal of the average response time as the assigned reward rate. For each configuration \( (n_0, n_1, n_2) \) of the failure/repair model, the queueing network model is solved for the throughput \( T(n_0, n_1, n_2) \) and average response time \( R(n_0, n_1, n_2) \) using SHARPE. The reward functions in the failure/repair model are then \( T(n_0, n_1, n_2) \) and \( 1/R(n_0, n_1, n_2) \).

Figures 12 & 13 show the mean reward (throughput and reciprocal response time) rates of the 3 configurations. When the processing capability, memory, and I/O devices were doubled, the number of failure points also increased — affecting the performability results. To verify this, figure 14 compares the reliability of the 3 systems with configuration \( (8,4,4) \) with improvement reliability with respect to \( (16,8,8) \) during initial operational periods. Figure 15 shows the mean cumulative reward (system throughput). Results of a fourth reconfiguration \( (16,4,4) \) show that simply doubling the computational capacity of the
system — with respect to configuration (8,4,4) — improves neither the system reliability nor its performability. On the contrary, the decrease corresponds to the combined effect of increased failure leaks due to imperfect coverage (\(c = 0.99\)). Even under perfect coverage, an improvement in performability is minimal as long as the number of I/O units which contribute the most to system failure remains the same.

REFERENCES


AUTHORS

Dr. Noe Lopez-Benitez; Dept. of Computer Science; Texas Tech Univ; Lubbock, Texas 79409-3104 USA.

Noe Lopez-Benitez received the BS in Communications & Electronics from the University of Guadalajara, Guadalajara in 1976, the MS in Electrical Engineering from the University of Kentucky in 1980, and the PhD in Electrical Engineering from Purdue University in 1989. From 1980 to 1983 he was with the IIE (Electrical Research Institute) in Cuernavaca, Mexico. He is on the Faculty of the Dept. of Computer Science at Texas Tech University, Lubbock. His research interests include fault-tolerant computing systems, reliability, and performance modeling. He is a member of the IEEE, ACM, and The Society for Computer Simulation.

Dr. Kishor S. Trivedi; Dept. of Electrical Eng'g; Box 9029; Duke Univ; Durham, North Carolina 27708-0291 USA

Kishor S. Trivedi holds a PhD in Computer Science from the University of Illinois, Urbana-Champaign. He is the author of the text, Probability and Statistics with Reliability, Queuing and Computer Science Applications. His research interests are in reliability and performance evaluation. Dr. Trivedi has lectured and published extensively on these topics. He is a Fellow of the IEEE and a Professor of Electrical Engineering and Computer Science at Duke University. He was an Editor of the IEEE Transactions on Computers from 1983-1987 and a co-designer of HARP, SAVE, SHARPE, and SPNP modeling packages.

No manuscript TR91-021 received 1991 February 20; revised 1992 November 21.

IEEE Log Number 07329