

Composite performance and dependability analysis

Kishor S. Trivedi, Jogesh K. Muppala *

Dept. of Electrical Engineering, Duke University Durham, NC 27706, USA

Steven P. Woollet

IBM, Research Triangle Park, NC 27709, USA

Boudewijn R. Haverkort

Tele-Informatics and Open Systems, University of Twente, 7500 AE Enschede, The Netherlands

Abstract

Trivedi, K.S., J.M. Muppala, S.P. Woollet and B.R. Haverkort, Composite performance and dependability analysis, Performance Evaluation 14 (1992) 197–215.

Composite performance and dependability analysis is gaining importance in the design of complex, fault-tolerant systems. Markov reward models are most commonly used for this purpose. In this paper, an introduction to Markov reward models including solution techniques and application examples is presented. Extensions of Markov reward models to semi-Markov reward models are also mentioned. A brief discussion of how task completion time models and models of queues with breakdowns and repairs relate to Markov reward models is also given.

Keywords: Markov chains, Markov reward models, performance, dependability, fault-tolerant systems, numerical techniques.

1. Introduction

Dependable fault-tolerant computer systems (FTCSs) are designed to continue their operation even in the presence of component or sub-system failures albeit at a reduced performance level. For example in spacecraft flight-control systems, it is essential that computations meet certain real-time constraints so as to avoid catastrophic failures. System outage even for a short period of time cannot be tolerated. Thus it is necessary to design these systems with sufficient redundancy so as to ensure that the performance criteria continue to be met despite faults in certain modules. On the other hand, in transaction processing systems, it is essential that certain throughput and response time requirements be satisfied. Minor interruptions in service, in these cases, are however tolerable.

Analysis of FTCSs from the pure performance viewpoint tends to be optimistic since it ignores the failure–repair behavior of these systems. On the other hand, pure dependability¹ analysis tends to be too conservative, since performance considerations are not taken into account. Thus it is essential that methods for the combined evaluation of performance and dependability be developed.

The first approach is to combine the performance and dependability behavior into an exact monolithic model. Two distinct problems arise from this approach, namely, *largeness* and *stiffness*. The largeness problem can be alleviated to some extent by using automated methods for

¹ The term *dependability* as an all-encompassing definition for reliability, availability, safety and security was introduced by Laprie [34].

* Now with Software Productivity Consortium, Herndon, VA.

generating the Markov chains and using efficient methods of storing the large state space, by exploiting the sparsity of the Markov chain generator matrix. Stiffness is caused in the overall model due to the large difference in the rates of occurrence of performance-related events and the rates of the rare, failure-related events. Stiffness leads

to difficulty in the solution of the model and numerical instability. A method of dealing with stiffness through aggregation/disaggregation has been proposed [4].

An alternative to the monolithic approach is as follows. When we examine the failure-repair and the performance behavior of FTCSs closely, we



Kishor S. Trivedi received the B. Tech. Degree from the Indian Institute of Technology, Bombay, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana-Champaign.

He is the author of a widely used text, *Probability and Statistics with Reliability, Queueing and Computer Science Applications* (Prentice-Hall, Englewood Cliffs, NJ). Both the text and his related research activities have been focused on computing system reliability and performance evaluation. Presently, he is a Professor of Electrical Engineering and Computer Science at Duke University, Durham, NC. He has served as a Principal Investigator on various AFOSR, ARO, Burroughs, IBM, NASA, NIH, NSF, ONR, and SPC funded projects and as a consultant to the industry and research laboratories.

Dr. Trivedi was an Editor of the *IEEE Transactions on Computers* from 1983 to 1987. He was an editor of the *Journal of Parallel and Distributed Computing*. He is a codesigner of HARP, SAVE, SHARPE, and SPNP packages. He is a fellow of the IEEE.



Jogesh K. Muppala received the Ph.D. degree in electrical engineering from Duke University, Durham, NC in 1991, the M.S. degree in computer engineering from The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA in 1987 and the B.E. degree in electronics and communication engineering from Osmania University, Hyderabad, India in 1985. He is currently a member of the technical staff at Software Productivity Consortium, Herndon, VA, USA. His research interests include stochastic Petri nets, performance and dependability modeling and multiprocessor systems. Dr. Muppala is a member of IEEE, IEEE computer society and ACM.



Steven Woollet received bachelors degrees in mathematics and electrical engineering from Bethel College (Indiana) and the University of Notre Dame, respectively, and his M.S. in electrical engineering from the University of Minnesota. He is currently working towards his Ph.D. in electrical engineering at Duke University, Durham, NC.

Mr. Woollet joined IBM in 1980 at Rochester, MN where he was involved in the development of terminals. He later moved to the Research Triangle Park, NC location where he is currently working on system performance analysis. His research interests include reliability, performance and performability modeling of networks and multiprocessor systems. Mr. Woollet is a member of IEEE, IEEE Computer Society and ACM.



Boudewijn R. Haverkort was born in Lichtenvoorde, the Netherlands in 1964. He received an M.Sc. and a Ph.D. degree in computer science, both from the University of Twente, Enschede, the Netherlands, in 1986 and 1991, respectively.

In 1986 he stayed half a year at the University of Dortmund, West-Germany, where he worked on the performance modelling of local-area network interconnection structures.

Since January 1990, Dr. Haverkort is an assistant professor in the Tele-Informatics and Open Systems group of the Department of Computer Science at the University of Twente, where he teaches an undergraduate course on performance analysis of communication networks.

His current research interests include performance, dependability and performability evaluation of fault-tolerant and distributed computer/communication systems, software tools for these evaluations, as well as formal specification techniques (Lotos) and their integration with performability analysis.

Dr. Haverkort, who is a member of the IEEE Computer Society and the ACM, received a Koninklijke Shell research prize early 1991.

notice that the failure and repair events are rare, i.e., the rate of occurrence of these events is very small compared with the rates of the performance-related events. Consequently, we can assume that the system attains a (quasi-)steady state with respect to the performance related events between successive occurrences of failure-repair events. Thus, we can compute the performance measures for the system in each of these (quasi-)steady states. The overall system can then be characterized by weighting these quasi-steady state performance measures by the structure state probabilities. This leads to a natural hierarchy of models: a higher level dependability model and a set of lower level performance related models, one for every state in the dependability model.

Several authors have used the latter concept in developing techniques for combined performance and dependability analysis of FTCSs. Early and defining work in this field was done by Beaudry [2] who computed the *computational availability* until failure for a computer system. Meyer [37,38] proposed the framework of *performability* for modeling fault-tolerant systems. All these models can be brought under the broad framework of Markov reward models (MRM) which is the emphasis of this paper.

In Section 2 we present a brief introduction to Markov reward models and the corresponding measures that can be computed. We illustrate these concepts and measures through several examples in Section 3. A brief discussion on the solution methods for MRMs is given in Section 4 and some extensions to the MRM model class are presented in Section 5. In this section we relate Markov reward models to task completion time problems and queues with breakdowns. We also present a brief discussion on generation methods for Markov reward models in Section 6. Some concluding remarks are given in Section 7.

2. Markov reward models: definition and measures

In this section we present a formal definition of Markov reward models and discuss the measures that can be obtained from them. First, we consider the formal definition of Markov chains.

Let $\{Z(t), t \geq 0\}$ represent a homogeneous finite-state continuous time Markov chain (CTMC) with state space Ω . Let N be the number of states in the Markov chain. The generator matrix is given by $Q = [q_{ij}]$ where q_{ij} , ($i \neq j$) represents the transition rate from state i to state j and the diagonal elements $q_{ii} = -q_i = -\sum_{j \neq i} q_{ij}$. Let $P_i(t)$ be the unconditional probability of the CTMC being in state i at time t , then the row vector $P(t)$ represents the *transient state probability vector* of the CTMC. The behavior of the CTMC can be described by the following Kolmogorov differential equation:

$$\dot{P}(t) = P(t)Q, \quad P(0) = p_0, \quad (1)$$

where $\dot{P}(t) = d(P(t))/dt$ and p_0 represents the initial probability vector of the CTMC. Let π_i be the steady-state probability of state i of the CTMC and π be the steady-state probability vector. In steady state we know that $\dot{P}(t) = 0$. By substituting this in eq. (1) we can derive the following equations for the steady state probabilities:

$$\pi Q = 0, \quad \sum_{i \in \Omega} \pi_i = 1. \quad (2)$$

Define $L(t) = \int_0^t P(u) du$. Then $L_i(t)$ is the expected total time spent by the CTMC in state i during the interval $[0, t)$. $L(t)$ satisfies the differential equation:

$$\dot{L}(t) = L(t)Q + p_0, \quad L(0) = \mathbf{0}, \quad (3)$$

which is obtained by integrating eq. (1).

For a Markov chain with absorbing states, computation of the mean time to absorption is of interest. Let A represent the set of absorbing states and $B = \Omega - A$ the set of the transient states in the CTMC. From the matrix Q a new matrix Q_B of size $|B| \times |B|$, where $|B|$ is the cardinality of the set B , can be constructed by restricting Q to only those states in B . Let $z_i = \int_0^\infty P_i(\tau) d\tau$, $i \in B$. Here z_i represents the mean time spent by the CTMC in state i until absorption. The row vector $z = [z_i]$ satisfies the following equation:

$$zQ_B = -P_B(0) \quad (4)$$

The above equation can be obtained by taking the limit $t \rightarrow \infty$ of eq. (3), with $z = L_B(\infty)$ and

noting that $\dot{L}_B(\infty) = 0$. The mean time to absorption MTTA is then computed as

$$MTTA = \sum_{i \in B} z_i.$$

Markov chains have been extended by assigning rewards to the states as well as to the transitions [26]. In the former case we speak of rate-based MRMs; in the latter case we speak of impulse-based MRMs. Combinations of the two types are of course also possible. In this paper we restrict ourselves to the former type. Let us now define a reward rate vector r over the states of the CTMC such that a reward rate r_i is associated with state i . A reward of $r_i \tau_i$ is accumulated when the sojourn time of the process in state i is τ_i . Let $X(t)$ represent the instantaneous reward rate of the MRM, then $X(t) = r_{Z(t)}$. Let $Y(t)$ denote the accumulated reward in the interval $[0, t)$. Then $Y(t)$ is given by:

$$Y(t) = \int_0^t X(\tau) d\tau = \int_0^t r_{Z(\tau)} d\tau.$$

The expected instantaneous reward rate $E[X(t)]$, the expected accumulated reward $E[Y(t)]$ and the steady-state expected reward rate $E[X] = E[X(\infty)]$ can be computed as

$$E[X(t)] = \sum_{i \in \Omega} r_i P_i(t),$$

$$E[Y(t)] = \sum_{i \in \Omega} r_i \int_0^t P_i(\tau) d\tau = \sum_{i \in \Omega} r_i L_i(t),$$

and

$$E[X] = \sum_{i \in \Omega} r_i \pi_i$$

respectively. For a Markov chain with absorbing states, the expected accumulated reward until absorption $E[Y(\infty)]$ can be computed as,

$$E[Y(\infty)] = \sum_{i \in \Omega} r_i \int_0^\infty P_i(\tau) d\tau = \sum_{i \in B} r_i z_i.$$

The distribution of $X(t)$ can be computed easily as

$$P[X(t) \leq x] = \sum_{r_i \leq x, i \in \Omega} P_i(t).$$

Beaudry [2] describes a method for computing $P[Y(\infty) \leq y]$, the distribution of accumulated reward until absorption, assuming that all non-absorbing states have positive reward rates assigned

Table 1
A look at reward based measures

Measure	Author(s)
$E[X]$	Gay & Ketelsen [17] Kubat [30] Muppala & Trivedi [41] Trivedi et al. [58]
$E[X(t)]$	Blake et al. [3] Das & Bhuyan [13] Heimann et al. [25] Muppala & Trivedi [41] Najjar & Gaudiot [44]
$E[Y(t)]$	Blake et al. [3] Najjar & Gaudiot [44]
$P[X(t) = x]$	Huslende [27] Levy & Wirth [35] Wu [59]
$P[Y(\infty) \leq y]$	Beaudry [2] Ciardo et al. [9]
$P[Y(t) \leq y]$	Ciciani & Grassi [12] De Souza & Gail [14] Donatiello & Iyer [15] Goyal & Tantawi [19] Meyer [38] Smith et al. [53]

to them. Ciardo et al. [9] extend this method to allow for transient states with zero rewards and also allow the system to be a semi-Markov process.

The distribution of the accumulated reward over a finite horizon, $\mathcal{Y}(y, t) = P[Y(t) \leq y]$, on the other hand is difficult to compute. $\mathcal{Y}(y, t)$ satisfies the equation,

$$(sI + uR - Q)\tilde{\mathcal{Y}}^*(u, s) = e.$$

Here $\tilde{\mathcal{Y}}^*(u, s)$ is $\mathcal{Y}(y, t)$ with the Laplace-Stieltjes transform () taken with respect to y followed by a Laplace transform (*) taken with respect to t , I is an identity matrix, $R = \text{diag}[r_1, r_2, \dots, r_N]$, is a diagonal matrix of reward rates and e is a vector whose entries are all equal to 1. Numerical methods for computing the distribution are presented in [14,15,19,38,47,53]. In Table 1 we list different reward based measures that have been computed in the past.

Given the MRM framework, the next immediate question that arises is “what are the appropriate reward rate assignments?” The reward rate vector to be assigned depends on whether we are

interested in performance, dependability or composite performance and dependability measures. For example, to compute the unreliability of a system, we consider a Markov chain in which all the system failure states are absorbing. We attach a reward rate of 1 to all the down states and zero to all the remaining states. Then $E[X(t)]$ gives the unreliability of the system at time t . By assigning a reward rate of 1 to all the up states and zero to all the down states, $E[X(t)]$ yields the reliability while $E[Y(\infty)]$ gives the mean time to failure of the system. To compute the availability of the system, define a reward rate vector associating a reward rate of 1 with all the up states and a reward rate of zero with all the down states. The instantaneous availability is given by $E[X(t)]$, the total uptime in the interval $[0, t)$ is given by $E[Y(t)]$ and the interval availability is $1/t E[Y(t)]$. It should be pointed out that reliability and mean time to system failure are meaningful measures only when all the system down states are absorbing states. Conversely, steady-state availability is meaningful only if none of the system states are absorbing states. Instantaneous availability, on the other hand, can be computed in any case.

Table 2
A look at various performance measures used as rewards

Reward rate	Author(s)
Bandwidth	Blake et al. [3] Das & Bhuyan [13] Smith et al. [53]
Number of processors	Najjar & Gaudiot [44]
Throughput	Gay & Ketelsen [17] Meyer [37,38]
Simultaneously active processors	Grassi et al. [20]
Number of transponders	Ciciani & Grassi [12]
Frequency of over-tolerance outages	Heimann et al. [25] Trivedi et al. [58]
Task interruption probability	Heimann et al. [25] Trivedi et al. [58]
Mean response time	Wu [59]
Response time distribution	Muppala & Trivedi [41] Muppala et al. [43] Reibman [48] Trivedi et al. [58]

Pure performance measures can also be computed using the same framework. For example, if we model the behavior of a queue using the underlying Markov model, the expected number of customers waiting in the queue can be computed by assigning the number of customers at the queue in state i of the Markov model as the reward rate in that state. The expected throughput at the queue can also be computed by assigning the rate of the transition from state i corresponding to the departures from the queue as the reward rate in that state.

If the reward rates are defined to reflect the performance levels of the system in the different configurations, then measures like the expected total amount of work completed in the interval $[0, t)$, the expected throughput of the system with failures and repairs, etc., can be computed. A related approach is to decide the assignment of reward rates based on some performance threshold [35]. We can define the threshold on a performance index and designate all states in which this performance index is below the threshold to be down states (assign a reward rate of zero) and the remaining states as up states (reward state of 1). This approach is well suited for degradable computer systems where, more often than not, the system is not completely unavailable due to failures, but its performance tends to degrade. In Table 2 we list the various performance measures that have been used as reward rates.

3. Markov reward models: examples

In this section we will look at some examples to illustrate the use of Markov reward models for the composite performance and dependability analysis of computer systems.

3.1. A multiprocessor system

We begin with an example given in [58]. Consider a multiprocessor system with n processors. Assume that the failure rate of each processor is γ . A processor failure is covered with probability c and is not covered with probability $\bar{c} = 1 - c$ [6]. Subsequent to a covered failure, the system comes

up in a degraded mode after a brief reconfiguration delay while after an uncovered failure, a longer reboot action is required. The reconfiguration times are assumed to be exponentially distributed with mean $1/\delta$, the reboot times are exponentially distributed with mean $1/\beta$, and the repair times are exponentially distributed with mean $1/\tau$.

It is assumed that no other event can take place during a reconfiguration or a reboot. The justification for this assumption lies in the fact that in practice the reconfiguration and reboot times are extremely small compared to the time between failures and the repair time. Assume that all failure events are mutually independent, and that a single repair facility is shared by all the processors. We are interested in computing measures based on the *steady-state* expected reward rate $E[X]$, for this example.

3.1.1. Markov model

The Markov model for this system is shown in Fig. 1. In state $i \in S_p = \{i | 1 \leq i \leq n\}$, the system is up with i processors functioning, and $n - i$ processors waiting for on-line repair. In state $x_i \in S_{rc} = \{x_{n-i} | i = 0, \dots, n - 2\}$, the system is undergoing a reconfiguration from a state with i operational processors to a state with $i - 1$ operational processors. In state $y_i \in S_{rb} = \{y_{n-i} | i = 0, \dots, n - 2\}$, the system is being rebooted from a state with i operational processors to a state with $i - 1$ operational processors. In state set $S_e = \{0\}$, the system is down and waiting for (off-line) repair.

Solving for the steady-state probabilities [58], we have

$$\pi_{n-i} = \frac{n!}{(n-i)!} (\gamma/\tau)^i \pi_n,$$

$$i = 0, 1, \dots, n,$$

$$\pi_{x_{n-i}} = \frac{n!}{(n-i)!} \frac{\gamma(n-i)c}{\delta} (\gamma/\tau)^i \pi_n,$$

$$i = 0, 1, \dots, n - 2,$$

$$\pi_{y_{n-i}} = \frac{n!}{(n-i)!} \frac{\gamma(n-i)(1-c)}{\beta} (\gamma/\tau)^i \pi_n,$$

$$i = 0, 1, \dots, n - 2,$$

where

$$\pi_n = \left[\sum_{i=0}^n (\gamma/\tau)^i \frac{n!}{(n-i)!} + \sum_{i=0}^{n-2} (\gamma/\tau)^i \frac{\gamma(n-i)cn!}{\delta(n-i)!} + \sum_{i=0}^{n-2} (\gamma/\tau)^i \frac{\gamma(n-i)(1-c)n!}{\beta(n-i)!} \right]^{-1}.$$

3.1.2. Reward assignment

It is assumed that the system is down while a reconfiguration, reboot or an off-line repair is in progress. The Unavailability U_s is then computed by assigning a reward rate of 1 to all the down states ($S_{rc} \cup S_{rb} \cup S_e$) and a reward rate of 0 to all the up states, i.e.

$$U_s = \sum_{i \in S_{rc}} \pi_i + \sum_{i \in S_{rb}} \pi_i + \sum_{i \in S_e} \pi_i. \tag{5}$$

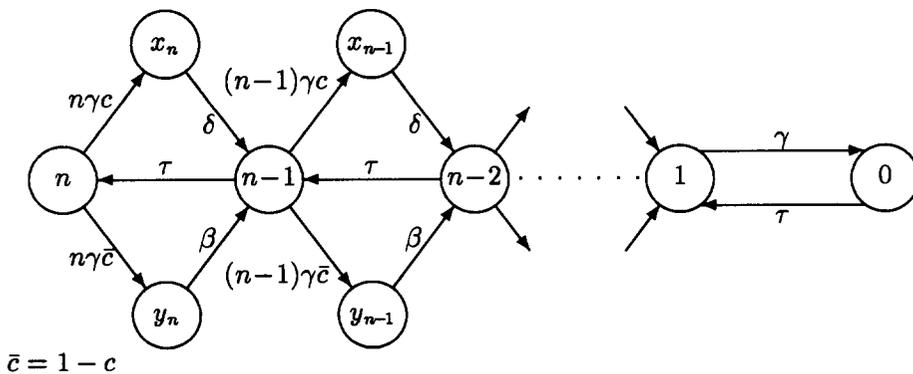


Fig. 1. Markov chain for computing the dependability of a multiprocessor system.

Assume that tasks arriving to the system form a Poisson process with rate λ and that the service requirements of tasks are independent, identically distributed according to an exponential distribution with mean $1/\mu$. It is assumed that there is a limited number b of buffers available for queueing the tasks. Tasks arriving when all the buffers are full, are rejected. An $M/M/i/b$ model is used to compute the probability $q_b(i)$ of a task being rejected because the buffers are full [22]:

$$q_b(i) = \begin{cases} \frac{\rho^b}{i^{b-i}i!} \left[\sum_{j=0}^{i-1} \frac{\rho^j}{j!} + \sum_{j=i}^b \frac{\rho^j}{i^{j-i}i!} \right]^{-1}, & b \geq i, \\ \frac{\rho^b}{b!} \left[\sum_{j=0}^b \frac{\rho^j}{j!} \right]^{-1}, & b < i, \end{cases} \quad (6)$$

where $\rho = \lambda/\mu$.

Arriving tasks are always rejected when the system is down. The *normalized throughput loss*, NTL, defined as the fraction of the jobs rejected, is obtained by assigning a reward rate to each state, equal to the probability that a task is rejected in that state. For the up states, this amounts to assigning a reward rate of $q_b(i)$, since the probability that a job is rejected in these states is equal to the probability that all the buffers are full. For all the down states the reward rate is 1,

since an arriving job is always rejected when the system is unavailable. Thus,

$$\begin{aligned} \text{NTL} &= \sum_{i \in S_p} q_b(i)\pi_i + \sum_{i \in S_{rc}} \pi_i + \sum_{i \in S_{rb}} \pi_i + \sum_{i \in S_e} \pi_i \\ &= \sum_{i \in S_p} q_b(i)\pi_i + U_s \end{aligned} \quad (7)$$

Equation 7 is basically the complementary version of Meyer's normalized throughput [38]. Both measures considered above are examples of *steady-state* expected reward rate $E[X]$.

3.1.3. Numerical results

We now assign some numerical values and evaluate the model. Let the failure rate γ of the processors be 1/6000 failures per hour. The mean time to reboot $1/\beta$ is 5 min and the mean time to repair a processor $1/\tau$ equals 1 h. The reconfiguration is accomplished in mean time $1/\delta = 10$ s. We assume jobs arrive at a rate of λ , where λ is varied from 0 to 200 jobs per second. The normalized throughput loss evaluated at $\lambda = 0$ is equivalent to the unavailability. Each processor has a service rate of $\mu = 100$ jobs per second.

Figure 2 shows the values of the unavailability and normalized throughput loss with the number of processors varying from 1 to 10. The coverage factor c is set to 0.98 and the number of buffers, b , is set to 10. Noting that the difference between the unavailability U_s and the normalized through-

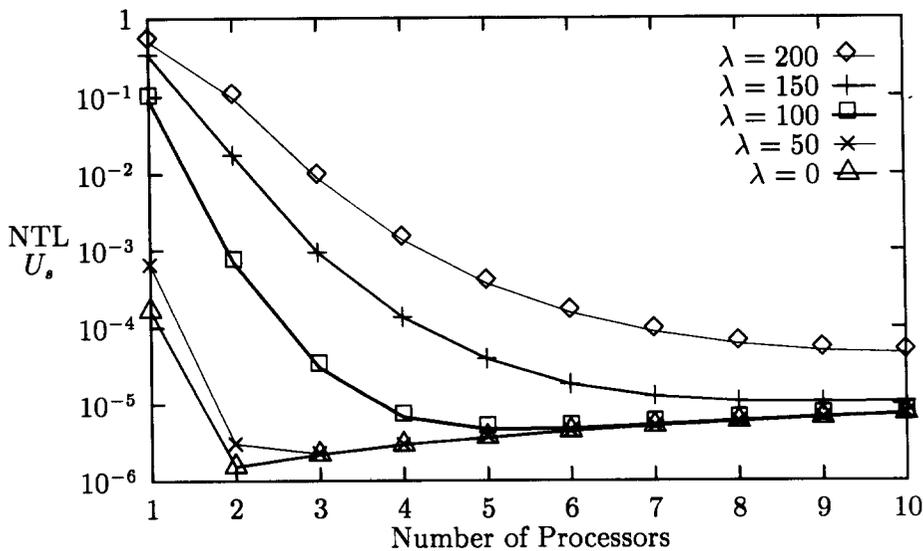


Fig. 2. Unavailability ($\lambda = 0$) and normalized throughput loss.

put loss NTL is only caused by the lack of buffer space (the term $\sum_{i \in S_p} q_b(i) \pi_i$ in Eq. 7), it can be observed that the lack of buffer space is a significant factor in job loss for systems having fewer than 4 processors for small values of λ ; as λ is increased, this factor is apparent for systems across the range of processors.

3.2. A multiple-bus multiprocessor system

Das and Bhuyan [13] consider a multiprocessor system using a multiple-bus interconnection. The multiprocessor has M processors, N memories, and B buses where $B \leq \min(M, N)$. All the processors and memories are connected to every bus. An arbiter handles the allocation of a bus to a memory having an outstanding request. In this example, we illustrate the use of expected instantaneous reward rate at time t , $E[X(t)]$.

3.2.1. Markov model

Due to the size and complexity of the Markov chain for this example, it is not given in this paper. Interested readers are referred to the original paper [13]. Failure times are assumed to be exponentially distributed with rates λ_p , λ_m , and λ_b for the processors, memories and buses, respectively. The failure coverage for the example

considered is assumed to be 1. The reliability of this system is found by summing the probabilities of being in the nonfailed states, i.e.

$$R(t) = \sum_{(j1, j2, j3) \in UP} P_{(j1, j2, j3)}(t),$$

where $(j1, j2, j3)$ represents a state with $j1$ processors, $j2$ memories and $j3$ buses functioning. For the system to be functioning, at least I processors and I memories must be functioning and be able to communicate with each other.

3.2.2. Reward assignment

The performance metric of consideration is available bandwidth. This bandwidth is defined as

$$BW_{(M, N, B)} = N\alpha - \sum_{i=B+1}^N (i - B) \times q(i),$$

for a system with M processors, N memories and B buses. Here α is defined as the probability of having at least one request for memory module i , and $q(i)$ is the probability of having exactly i memory requests in a cycle. For the case of uniform memory accesses (every processor has equal probability of requesting access to every memory) and assuming that each processor generates a request for memory in every cycle, and the number of processors M is equal to the

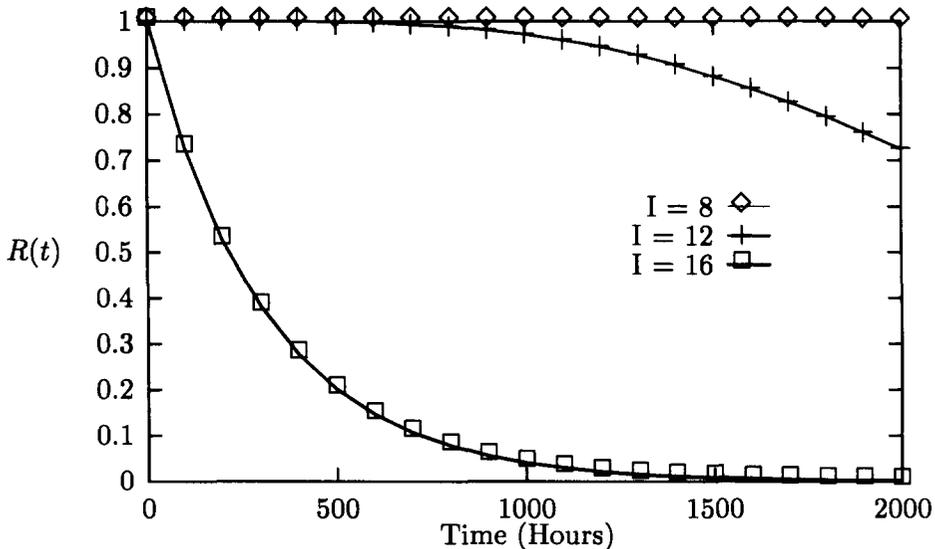


Fig. 3. Reliability for 16x16x8 multiple-bus multiprocessor requiring I processors and I memories.

number of memories N , these variables can be computed by

$$\alpha = 1 - (1 - (1/N))^N,$$

and

$$q(i) = \binom{N}{i} \alpha^i (1 - \alpha)^{N-i}.$$

To determine the bandwidth availability, i.e. the expected instantaneous bandwidth $BA(t)$, we assign a reward rate of $BW_{(j1, j2, j3)}$ to each non-failed state $(j1, j2, j3)$ and 0 to all the failed states. Then,

$$BA(t) = \sum_{(j1, j2, j3) \in UP} BW_{(j1, j2, j3)} P_{(j1, j2, j3)}(t).$$

Note that this measure is an example of the expected instantaneous reward rate at time t , $E[X(t)]$.

3.2.3. Numerical Results

Figures 3 and 4 show the reliability and bandwidth availability for a multiple-bus multiprocessor having $M = 16$ processors, $N = 16$ memories, and $B = 8$ buses for three different values of I , namely, 8, 12 and 16. The number of states in the Markov chain is 649 for $I = 8$, 201 for $I = 12$ and 9 for $I = 16$, respectively. The failure rates are assumed to be $\lambda_p = \lambda_m = 0.0001$ failures per hour, and $\lambda_b = 0.00005$ failures per hour. The system is considered operational as long as I processors

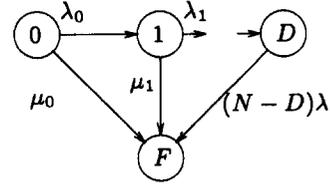


Fig. 5. Markov chain for dependability aspects of a hypercube-based multiprocessor system.

and I memories are functional and are able to communicate. From the figures we can notice the degradation both in terms of reliability as well as the bandwidth availability with increase in time. Although the reliability plot for $I = 8$ suggests that the system is quite reliable over the interval of time considered, the degradation in this case is more apparent when we consider the bandwidth availability. The bandwidth availability is reduced by more than 12% of its original value for $I = 8$ at $t = 2000$ h.

3.3. A hypercube-based multiprocessor system

In this example, we consider a multiprocessor system having an interconnect based on a hypercube topology, adapted from Najjar and Gaudiot [44]. For this example, we compute measures based on the expected instantaneous reward rate $E[X(t)]$ at time t , the expected accumulated reward $E[Y(t)]$ until time t and the distribution of

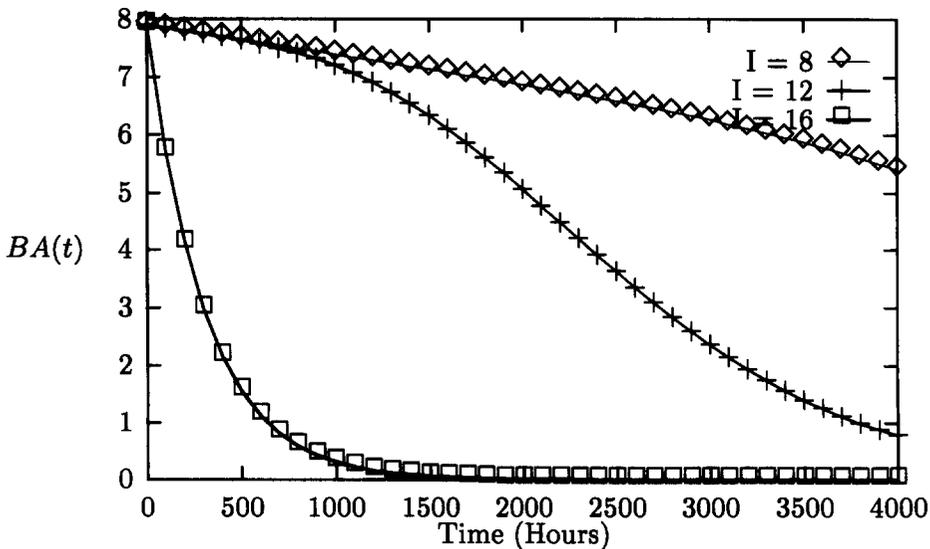


Fig. 4. Bandwidth availability for $16 \times 16 \times 8$ multiple-bus multiprocessor requiring I processors and I memories.

accumulated reward until system failure $P[Y(\infty) \leq y]$.

3.3.1. Markov model

The Markov chain for the failure process of the multiprocessor system is shown in Fig. 5. The state of the system is defined as the number of failed nodes or processors for states 0 through D . The state F is defined to be the *failed* state. We assume that no more than D failures can occur. This chain gives no indication of the connection topology. The method used to consider the disconnection of the system is to incorporate disconnection probabilities into the coverage factor.

The transition rates of the CTMC in Fig. 5 are defined in terms of the single processor failure rate λ as follows:

$$\lambda_i = C_i(N - i)\lambda,$$

and

$$\mu_i = (1 - C_i)(N - i)\lambda.$$

Note that the coverage factor C_i is state dependent. It is defined as $C_i = c(1 - Q^{(N)}(i))$, where c is the probability of successful recovery from a failure and $Q^{(N)}(i)$ is the probability of a disconnection occurring at the i^{th} node failure, given that no disconnection had occurred up to the $(i - 1)^{st}$ node failure in a system having $N = 2^n$ nodes. The cube dimension is defined by n .

In [44] it is shown that $Q^{(N)}(i)$ can be approximated by considering the probability of a single node disconnection for cubes having dimension $n \geq 6$. The probability of a single node disconnection is given by

$$Q^{(N)}(i) = \begin{cases} 0, & i < n, \\ \frac{N}{\binom{N}{n}}, & i = n, \\ \frac{nN \binom{N-n-1}{i-n}}{(N-i+1) \binom{N}{i-1}}, & i > n. \end{cases}$$

3.3.2. Reward assignment

For this example we will show plots for the reliability $R(t)$, the normalized computational availability $E[CA(t)]$, the expected number of processor hours $E[CW(t)]$ available until time t , and the complementary distribution of total pro-

cessor-hours until system failure $P[CW(\infty) > y]$. The first three measures are defined by:

$$R(t) = \sum_{i=0}^D P_i(t),$$

$$E[CA(t)] = \sum_{i=0}^D \frac{N-i}{N} P_i(t) \quad \text{and}$$

$$E[CW(t)] = \sum_{i=0}^D (N-i)L_i(t).$$

The distribution of total processor-hours is determined using the method described in Section 4.

For the four measures shown here the reward rate associated with the failure state, F , is 0. To compute the reliability we associate a reward rate 1 with all other states. This measure gives the probability of being in an up state at time t . For the normalized computational availability we associate a reward rate $(N - i)/N$ with state $i(0 \leq i \leq D)$. This gives a measure of the percentage of total system processing power which is available at time t . A reward rate equal to $N - i$ is associated with state $i(0 \leq i \leq D)$ to compute the expected number of processing hours accumulated by time t , or the distribution thereof. For the reward rate assignments the underlying assumption is that the processing power is proportional to the number of operational processors.

In this example $R(t)$ and $E[CA(t)]$ are instances of the expected instantaneous reward rate $E[X(t)]$ at time t , whereas $E[CW(t)]$ is an instance of the expected accumulated reward $E[Y(t)]$ until time t and $P[CW(\infty) > y]$ is an instance of the distribution of accumulated reward until system failure.

3.3.3. Numerical results

For this example, we assume that the probability of successful recovery, c , is 1. The number of node failures before system failure, D , is assumed to be half of the total number of nodes in the system, N .

Figure 6 shows the reliability for cubes of dimension 6, 8, and 10. It can be seen that for a fixed λ , the cubes of higher dimension have higher reliability at small values of t and lower reliabilities at larger values of t . This indicates that systems with more nodes will have a higher probability of being in a non-failed state shortly after system start-up, but as time goes by, the higher

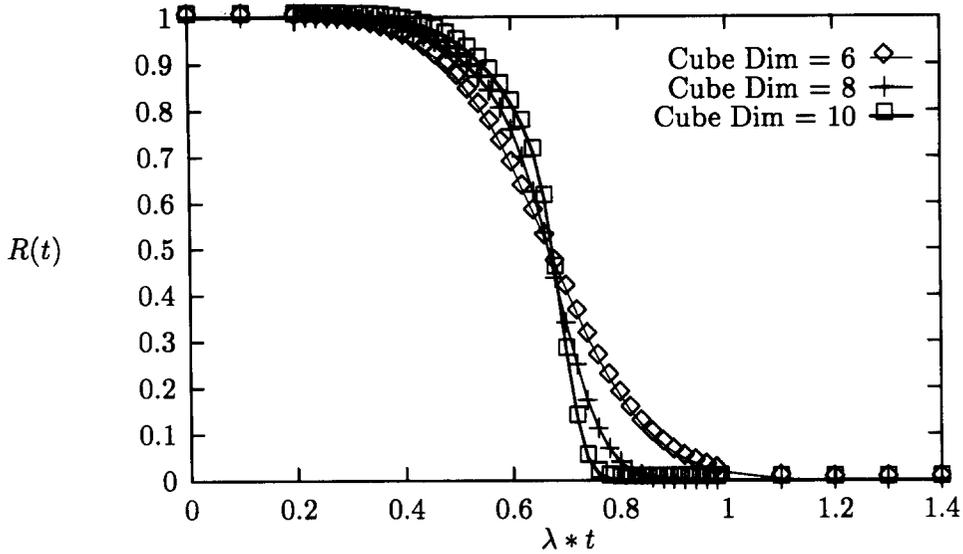


Fig. 6. Reliability for a log-connected hypercube-based multiprocessor system.

number of nodes will cause a greater chance of system failure.

Figure 7 indicates that in terms of *normalized* computational availability, all systems are very close for smaller values of t at a fixed λ . This is the percentage of total computational power for a given system. Again, it is shown that for larger values of t , the greater number of processors will cause a higher probability of being in the system failure state and a lower percentage of total processing power available.

Although $R(t)$ and $E[CA(t)]$ have been fairly similar for hypercube topologies with different cube dimensions, by examining Fig. 8, it can be observed that in terms of the expected amount of accumulated processor-hours, the larger system shows a significant advantage. Even though we have seen that a greater number of processors gives rise to more probability of failure for larger t , this figure indicates that the amount of work done in the time before failure is significantly higher for larger systems. The advantage of the

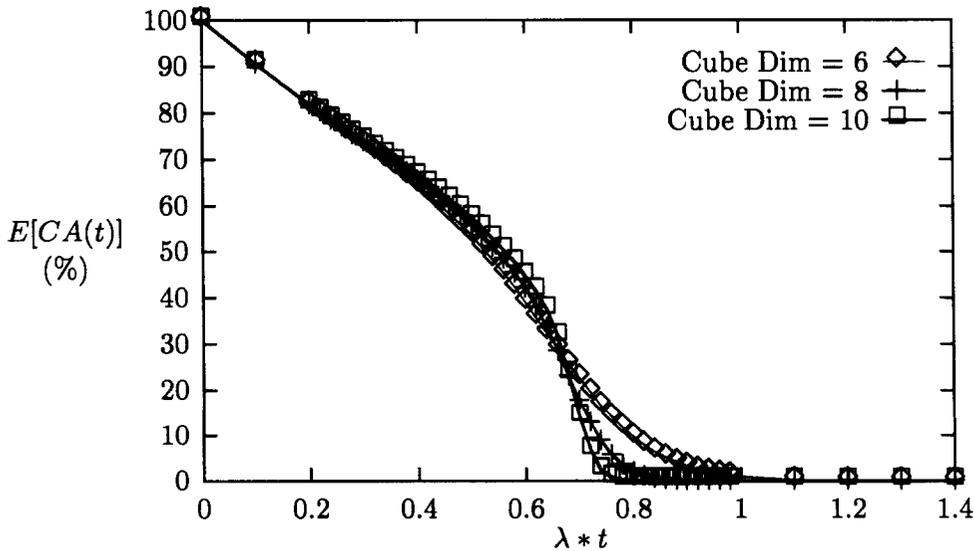


Fig. 7. Normalized computational availability for a log-connected hypercube-based multiprocessor system.

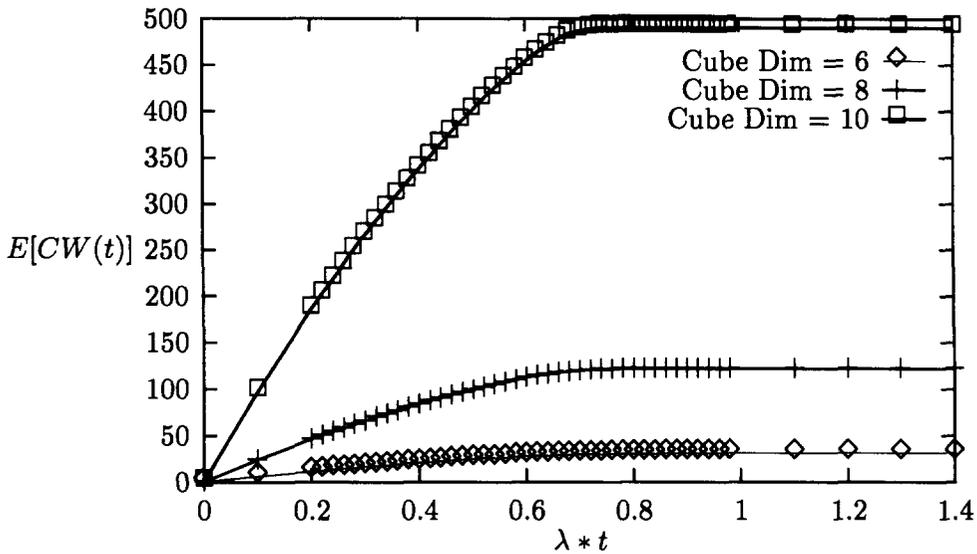


Fig. 8. Expected number of processor-hours available for a log-connected hypercube-based multiprocessor system.

larger cube is also quite evident when looking at the complementary distribution of accumulated processor-hours until system failure shown in Fig. 9. This figure indicates the probability that the total work accumulated by the system before failure (in processor-hours) is greater than y ($0 \leq y \leq 700$). For example, consider the probability that the total work accumulated by a system before failure will be greater than 100 processor-hours; a system having cube dimension of 6 has a probability close to zero, a system having cube dimen-

sion 8 has a probability of around 0.9 and a system of cube dimension 10 has almost 1.0 probability of doing 100 processor-hours worth of work.

4. Solution techniques for Markov reward models

In this section, we will briefly mention various methods available for solving Markov reward models. If we are interested only in the expected

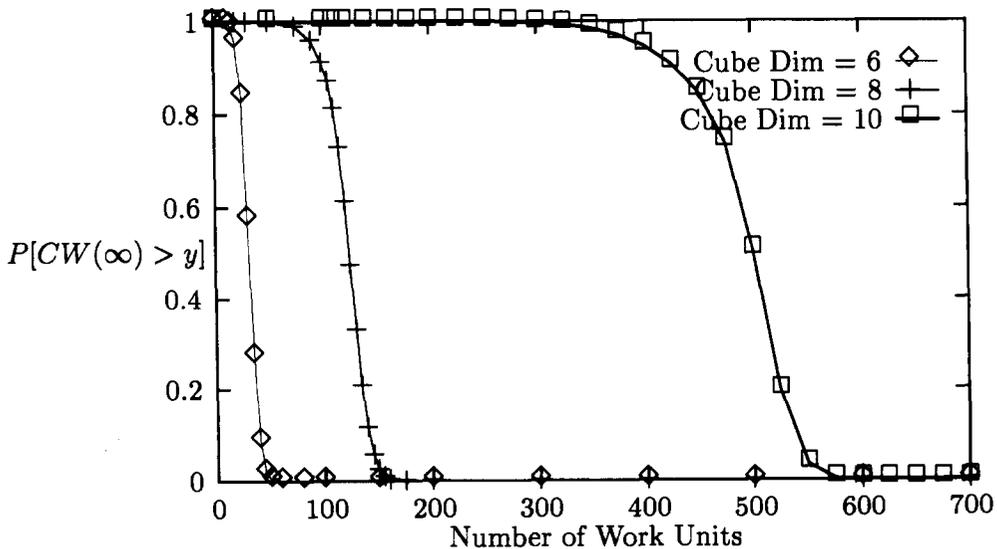


Fig. 9. Complementary distribution of the number of accumulated processor-hours until system failure for a log-connected hypercube-based multiprocessor system.

values of the reward-based measures, we need to obtain the state probabilities of the Markov chain.

The steady-state probabilities of the Markov chain can be obtained by solving the system of linear eqs. (2). Direct methods like Gaussian elimination [18,54] can be applied to solve these equations. However, these methods need full-storage for the matrix Q and the complexity is $O(N^3)$ where N is the number of states in the Markov chain. Thus it becomes prohibitively expensive to solve large Markov chains using these methods. In these cases, it is preferable to use iterative methods like Gauss–Seidel or Successive Overrelaxation (SOR) [54] for solution. These methods preserve the sparsity of the Q matrix and thus can be used to solve very large models. The iteration equation for SOR can be written as

$$\pi^{i+1} = \omega[\pi^{i+1}U + \pi^iL]D^{-1} + (1 - \omega)\pi^i,$$

where π^{i+1} is the solution vector at the i th iteration, L is a lower triangular matrix, U is an upper triangular matrix, D is a diagonal matrix such that $Q = D - L - U$ and $0 < \omega \leq 2$. Substituting $\omega = 1$ in the above equation yields the iteration equation for the Gauss–Seidel method. Near optimal assignment of ω has been discussed in [54] and has been used in [10].

A related problem is the computation of z using the system of linear eqs. (4). The discussion about direct and iterative solution methods is also applicable in this case.

The transient state probabilities, $P(t)$ can be obtained by solving the Kolmogorov differential eq. (1). Fully symbolic solution using Laplace transforms is possible only for Markov chains with a small number of states or if the Markov chain has a very regular structure [57]. Semi-symbolic solution of the Markov chains in terms of time t can be obtained via algebraic methods [56]. However, this algorithm is of $O(N^3)$ complexity, where N is the number of states in the Markov chain, and needs full matrix storage. This method can also be numerically unstable.

Numerical transient solution techniques, on the other hand, can take advantage of the sparsity of the Q matrix to reduce the space and time required for solution, and are numerically stable. We can use methods for solving linear differential equations like Runge–Kutta [49]. For stiff problems, an implicit Runge–Kutta method [36] is useful. Uniformization (also called randomiza-

tion) [49] is a very useful numerical method based on the subordination of a Markov chain to a Poisson process [23]. In uniformization, the matrix Q is transformed to obtain another matrix $Q^* = I + Q/q$, where $q \geq \max_i |q_{ii}|$. This transformation process is referred to as randomization. Then $P(t)$ is computed as,

$$\begin{aligned} P(t) &= \sum_{i=0}^{\infty} P(0)(Q^*)^i e^{-qt} \frac{(qt)^i}{i!} \\ &= \sum_{i=l}^k P(0)(Q^*)^i e^{-qt} \frac{(qt)^i}{i!} + \epsilon. \end{aligned}$$

The advantage of using uniformization is that a bound on the error due to truncation of the infinite series is known, i.e., we can pre-compute the lower truncation point l and the upper truncation point k such that the required error tolerance ϵ is satisfied. Furthermore, since the matrix Q^* contains only positive entries, numerical instability is avoided. Uniformization has been recently improved to handle stiff problems [42].

Solution of eq. (3) to compute $L(t)$ is similar to solving for $P(t)$. A uniformization based method for solving eq. (3) has been discussed in [50]. For a more detailed discussion of various transient solution methods, the reader is referred to [47].

To solve for $P\{Y(\infty) \leq y\}$, the distribution of the accumulated reward until absorption, the Markov chain Z is transformed in a new Markov chain Z' where the transition rate q'_{ij} from state i to state j in Z' (i is a transient state) is obtained by dividing the corresponding transition rate in Z by the reward associated with state i , i.e., $q'_{ij} = q_{ij}/r_i$. The transient solution of the Markov chain Z' yields the distribution of accumulated reward as

$$P\{Y(\infty) \leq y\} = \sum_{i \in A} P'_i(y),$$

where $P'_i(y)$ represents the transient state probability of state i at time y and A represents the set of absorbing states in the Markov chain Z' . This procedure was first described by Beaudry [2]. The above transformation is not applicable when the reward rate r_i associated with a transient state i is zero. Ciardo et al. [9] recently extended Beaudry's method to allow for transient states with zero reward rates.

Finally, as mentioned earlier, the computation of $P\{Y(t) \leq y\}$ is very difficult. Smith et al. [53] describe a method of computing this distribution using numerical inversion of a double Laplace transform. The algorithm is of complexity $O(N^3)$. Methods based on uniformization [14] and partial differential equations [46,47] have also been developed. Methods based on Laguerre transform techniques were presented in [28].

5. Markov reward models: extensions and related work

Some extensions to Markov reward models are discussed in Section 5.1. In Section 5.2 we show the relationship between MRMs and task completion time problems and in Section 5.3 we relate MRMs to models of queues with breakdowns.

5.1. Semi-Markov and non-homogeneous Markov reward models

Although we have restricted ourselves to Markov reward models in the previous sections of this paper, the concept of reward based modeling can easily be extended to semi-Markovian and non-homogeneous Markovian models. In [9] a semi-Markov reward model as well as methods for computing the distribution of the accumulated reward until absorption was proposed, thus extending the work of Beaudry [2]. Iyer et al. [29] derive Laplace transforms for the distribution of the accumulated reward assuming that the underlying process is semi-Markovian. Kulkarni et al. [32] also use semi-Markov processes in their performability related studies. Pattipati et al. [46] derived linear, hyperbolic partial differential equations describing the evolution of the accumulated reward, allowing the underlying Markov chain to be non-homogeneous and the reward rates to be time-dependent.

The MRMs can be extended to allow for rewards to be associated with the transitions, resulting in impulse-based MRMs. This has been considered in [11,52].

5.2. The completion time problem

So far we have assumed that whenever a state change occurs in the Markov chain, the reward

accumulated until that instant is preserved. We can relax this assumption and consider the possibility where such a state change results in loss of the accumulated reward. These models have been considered in the context of task completion time distributions [7,31,32]. Based on the loss behavior, the states of the underlying (semi)-Markov chain are classified into preemptive resume (PRS) and preemptive repeat (PRT) states. The accumulated reward is preserved upon transitions to PRS states whereas it is lost upon transitions to PRT states. When all the states are of PRS type, it has been proved that

$$P\{Y(t) \leq x\} = 1 - P\{T(x) < t\},$$

where $T(x)$ is the random variable representing the time to completion of a job with work requirement equal to x . Assuming a semi-Markovian server with the reward rates in state i being interpreted as the rate of service of the task being serviced and allowing the transitions among states to be either PRS or PRT type, Kulkarni et al. [32] derive the Laplace-Stieltjes transform $E[e^{-sT(x)}]$ of the task completion time. Numerical methods for inversion of the transform equations are discussed in [8]. An alternative approach based on phase-type expansion is presented in [5]. Extension to allow for checkpointing are considered in [16,33]. Using the completion time analysis as a basis, Nicola et al. [45] and Kulkarni et al. [33] also derive the equations for the mean response time, thus including the effects of queueing delays.

5.3. Queues with breakdown

In order to illustrate the connection between queues with breakdown and performability, we consider a finite-buffer $M/M/1/m$ queue with failure and repair of the server. The infinite buffer case was analyzed by Mitrani and Avi-Itzhak [40]. For this queueing system we assume that customers arrive at the queue at the rate λ . The service rate of the server is assumed to be μ . The server can fail with rate γ and is repaired with rate τ . Initially we assume that customers arriving while the server is down, are rejected.

First, we will consider the approximation of this queueing system using a two-level Markov reward model. The lower level model will be a

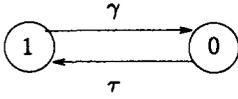


Fig. 10. Upper level dependability model for the $M/M/1/m$ system.

$M/M/1/m$ queue without failures. For this queue, the expected throughput $E[\hat{T}]$ is given as,

$$E[\hat{T}] = \frac{\mu\rho(1 - \rho^m)}{(1 - \rho^{m+1})},$$

where $\rho = \lambda/\mu$. The upper level dependability model consists of two states as shown in Fig. 10. In this figure, state 1 denotes that the server is functioning and state 0 denotes that the server has failed. For this Markov chain, the steady-state probabilities of the two states are given by

$$\pi_1 = \frac{\tau}{(\gamma + \tau)}, \quad \text{and} \quad \pi_0 = \frac{\gamma}{(\gamma + \tau)}.$$

To compute the steady-state expected throughput of the queue with failure and repair of the server, we assign a reward rate of $r_1 = E[\hat{T}]$ to state 1 and $r_0 = 0$ to state 0. The expected reward rate in steady-state, i.e.

$$E[T] = r_1\pi_1 + r_0\pi_0 = \frac{\mu\tau}{(\gamma + \tau)} \frac{\rho(1 - \rho^m)}{(1 - \rho^{m+1})}$$

yields the expected throughput with failures and repairs.

Now we will compute the throughput using the exact Markov model for the $M/M/1/m$ system with failures and repairs, in order to compare the accuracy of the approximation using an MRM. The overall Markov model for the queueing system is shown in Fig. 11. This is a very simple

Markov model whose steady-state solution is given as

$$\pi_{i,1} = \rho^i \frac{\rho\tau}{(\gamma + \tau)}, \quad \pi_{i,0} = \delta\rho^i \frac{\rho\tau}{(\gamma + \tau)},$$

where $\delta = \gamma/\tau$.

The expected throughput of this system can be computed as

$$E[T] = \sum_{i=1}^m \mu\pi_{i,1} = \frac{\mu\tau}{(\gamma + \tau)} \frac{\rho(1 - \rho^m)}{(1 - \rho^{m+1})}.$$

We see that the expected system throughput using an MRM is the same as the throughput computed using the exact model.

Finally, let us alter the model and allow for the accumulation of jobs while the server is down. Consequently, the exact Markov model for the system is modified by adding transitions from states $(i, 0)$ to $(i + 1, 0)$, $0 \leq i < m$ with rate λ . The modified Markov chain is shown in Fig. 12. It is extremely difficult to obtain the steady-state solution for this Markov chain in closed-form. Thus we resort to numerical methods for obtaining the steady-state probabilities $\pi_{i,j}$. The steady-state throughput for this altered system is then given by,

$$E[T_a] = \lambda \sum_{j=0}^1 \sum_{i=0}^{m-1} \pi_{i,j}.$$

The subscript a is used to indicate that this measure is for the *altered* system.

We will now consider an approximation for this system using the two-level Markov reward model shown in Fig. 10. We realize that it is difficult to reflect the effect of the arrivals while the server is down, using a reward rate attached to state 0. One possible approach is to ignore the effect of these arrivals, thereby attaching a reward rate of 0 to state 0. This leads us back to the

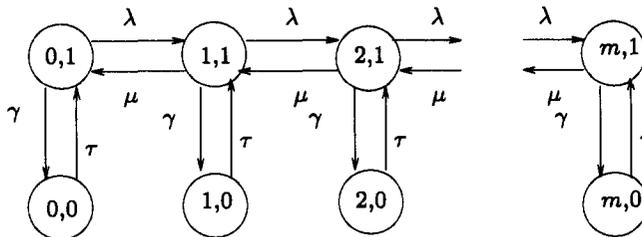


Fig. 11. Exact CTMC model of the $M/M/1/m$ system.

approximate MRM considered earlier. However, we must realize that the throughput obtained by solving this model will be smaller than the exact throughput, since it ignores the arrivals while the server is down. Thus, we get a lower bound on the throughput of this system as,

$$E[T_a]_{LB} = \frac{\lambda\tau}{(\gamma + \tau)} \frac{(1 - \rho^m)}{(1 - \rho^{m+1})}$$

Alternatively, since the arrival rate of the customers is λ , we could assign a reward rate of λ to state 0. However, this reward rate assignment implicitly assumes that arrivals can still occur when the buffer is full, *i.e.*, the system is in state $(m, 0)$ of the exact model. Thus the throughput computed by this model will be larger than the exact throughput. This will result in an upper bound on the exact throughput which is given by

$$E[T_a]_{UB} = \frac{\lambda\tau}{(\gamma + \tau)} \frac{(1 - \rho^m)}{(1 - \rho^{m+1})} + \frac{\lambda\gamma}{(\gamma + \tau)}$$

5.3.1. Numerical results

To illustrate this model further, we compute the exact throughput as well as the bounds on the throughput. We assume $\lambda = 60$ jobs per hour, $\mu = 120$ jobs per hour, $\gamma = 1/6000$ failures per hour and $\tau = 1$ repair per hour. The corresponding throughput values are shown in Table 3 as a function of m , the number of buffers. We note that as m increases, the upper bound gets closer to the exact value. This is to be expected, because as m increases, the probability of being in state $(m, 0)$ of the exact model decreases and thus the error caused due to assuming an arrival rate of λ in that state becomes smaller. In the limit, as m approaches infinity, the upper bound and the exact value will yield the same result. We furthermore note that as m increases, the lower bound

Table 3 Throughputs for the $M/M/1/m$ system

No. of buffers m	Exact $E[T_a]$	Upper bound $E[T_a]_{UB}$	Lower bound $E[T_a]_{LB}$
1	39.9934	40.0033	39.9933
2	51.4202	51.4300	51.4200
3	55.9909	56.0007	55.9907
4	58.0552	58.0648	58.0548
5	59.0383	59.0478	59.0378
6	59.5183	59.5276	59.5176
7	59.7555	59.7647	59.7547
8	59.8736	59.8826	59.8726
9	59.9325	59.9414	59.9314
10	59.9619	59.9701	59.9607
11	59.9767	59.9854	59.9754
12	59.9842	59.9927	59.9827
13	59.9880	59.9963	59.9863
14	59.9900	59.9982	59.9882
15	59.9910	59.9991	59.9891

diverges from the exact value. This is also to be expected since the larger m is, the more we ignore the effect of job accumulation when the server is down, *i.e.*, the more the throughput that we do not take into account.

5.3.2. Evaluation

The above example illustrates the fact that modeling using MRMs can sometimes lead to inaccurate results. Furthermore, it is difficult to model certain aspects of the state-dependent behavior of systems using MRMs, such as the building up of queues in temporarily unstable situations. We should note however, that the exact model will in general be large and stiff whereas the approximate approach involves solving two smaller and less stiff models. In many practical applications, generation and solution of an exact model is not feasible. Hence, we need to resort to approximation techniques.

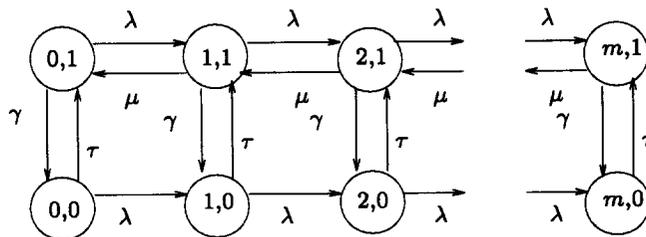


Fig. 12. Modified exact CTMC model of the $M/M/1/m$ system.

6. Generation techniques for Markov reward models

Earlier, we have alluded to the problem of generating the large Markov chains. Several authors have proposed methods of generating Markov chains automatically from a higher level description of the system. Stewart [55] describes a method of generating Markov chains using the concept of balls and buckets. Grassman [21] proposes a generation method based on events while Gross and Miller [23] use the SERT (State space, Events, Rate vectors, Target state vectors) approach.

Stochastic Petri nets (SPNs) (Ajmone Marsan et al. [1], Ciardo et al. [10].) provide an excellent high level interface to automatically generate the Markov chain. Ajmone Marsan et al. [1] describe a method of generating Markov chains from a generalized stochastic Petri net model. Meyer et al. [39,51] describe a method of computing performance using stochastic activity networks (SAN). Ciardo et al. [10] extend SPNs by associating rewards with the markings to obtain stochastic reward nets (SRN) [43].

7. Conclusions

In this paper we discussed the use of Markov reward models for composite performance and dependability analysis of fault-tolerant computer systems. Markov reward models as well as several interesting measures that can be obtained from them are discussed. The models and measures were illustrated using an interesting set of examples. Solution methods of MRMs were also briefly discussed. Various extensions of MRMs, such as semi-Markov chains and time-dependent transition rates were also mentioned. The use of MRMs for the composite analysis of performance and dependability of fault-tolerant computer systems was compared to models of queues with breakdowns. In order to use MRMs effectively, tools that help in automatically constructing them from a high-level description are important. Generation techniques for MRMs were briefly mentioned in this paper. We will deal with the gener-

ation techniques for MRMs in greater detail in a forthcoming paper [24].

References

- [1] M. Ajmone Marsan, G. Conte and G. Balbo, A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems, *ACM Trans. Comput. Syst.* **2** (2) (1984) 93–122.
- [2] M.D. Beaudry, Performance-related reliability measures for computing systems, *IEEE Trans. Comput.* **27** (6) (1978) 540–547.
- [3] J.T. Blake, A.L. Reibman and K.S. Trivedi, Sensitivity analysis of reliability and performance measures for multiprocessor systems, in: *Proc. 1988 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, Santa Fe, NM (1988) 177–186.
- [4] A. Bobbio and K.S. Trivedi, An aggregation technique for the transient analysis of stiff Markov chains, *IEEE Trans. Comput.* **35** (9) (1986) 803–814.
- [5] A. Bobbio and K.S. Trivedi, Computation of the distribution of the completion time when the work requirement is a PH random variable, *Stochastic Models* **6** (1) (1990) 133–150.
- [6] W.G. Bouricius, W.C. Carter and P.R. Schneider, Reliability modeling techniques for self-repairing computer systems. in: *Proc. 24th National Conference of the ACM* (1969) 295–309.
- [7] X. Castillo and D.P. Siewiorek, A performance-reliability model for computing systems, in: *Proc. 10th International Symposium on Fault-Tolerant Computing* (IEEE Computer Society Press, Los Alamitos, CA, 1980) 187–192.
- [8] P.F. Chimento, System performance in a failure prone environment, Ph.D. Thesis, Department of Computer Science, Duke University, Durham, NC, 1988.
- [9] G. Ciardo, R. Marie, B. Sericola and K.S. Trivedi, Performance analysis using semi-Markov reward processes, *IEEE Trans. Comput.* **39** (10) (1990) 1251–1264.
- [10] G. Ciardo, J. Muppala and K. Trivedi, SPNP: stochastic Petri net package, in: *Proc. International Workshop on Petri Nets and Performance Models* (IEEE Computer Society Press, Los Alamos, CA, 1989) 142–150.
- [11] G. Ciardo, J.K. Muppala and K.S. Trivedi, On the solution of GSPN reward models, *Performance Evaluation* **12** (4) (1991) 237–253.
- [12] B. Ciciani and V. Grassi, Performance evaluation of fault-tolerant satellite systems, *IEEE Trans. Comm.* **35** (4) (1987) 403–409.
- [13] C.R. Das and L.N. Bhuyan, Bandwidth availability of multiple-bus multiprocessors, *IEEE Trans. Comput.* **34** (10) (1985) 918–926.
- [14] E. de Souza e Silva and H.R. Gail, Calculating availability and performance measures of repairable computer systems using randomization, *J. ACM* **36** (1) (1989) 171–193.
- [15] L. Donatiello and B.R. Iyer, Analysis of a composite performance reliability measure for fault-tolerant systems, *J. ACM* **34** (4) (1987) 179–199.

- [16] A. Duda, The effects of checkpointing on program execution time, *Inform. Processing Letters* **16** (1983) 221–229.
- [17] F.A. Gay and M.L. Ketelsen, Performance evaluation for gracefully degrading systems, in: *Proc. IEEE International Symposium on Fault-Tolerant Computing FTCS-9* (IEEE Computer Society Press, Los Alamos, CA, 1979) 51–58.
- [18] A. Goyal, S.S. Lavenberg and K.S. Trivedi, Probabilistic modeling of computer system availability, *Ann. Oper. Res.* **8** (1987) 285–306.
- [19] A. Goyal and A.N. Tantawi, Evaluation of performability in acyclic Markov chains, *IEEE Trans. Comput.* **36** (6) (1987) 738–744.
- [20] V. Grassi, L. Donatiello and G. Iazeolla, Performability evaluation of multicomponent fault-tolerant systems, *IEEE Trans. Reliabil.* **37** (2) (1988) 216–222.
- [21] W.K. Grassmann, Markov modelling, in: S. Roberts, J. Banks and B. Schmeiser, eds., *Proc. of the 1983 Winter Simulation Conference* (1983) 613–620.
- [22] D. Gross and C.M. Harris, *Fundamentals of Queueing Theory* (John Wiley and Sons, New York, NY, 1974).
- [23] D. Gross and D. Miller, The randomization technique as a modeling tool and solution procedure for transient Markov processes, *Oper. Res.* **32** (2) (1984) 926–944.
- [24] B.R. Haverkort and K.S. Trivedi, Generation techniques for Markov reward models, in preparation.
- [25] D. Heimann, N. Mittal and K.S. Trivedi, Availability and reliability modeling of computer systems, in: M. Yovits, ed., *Advances in Computers, Vol. 31* (Academic Press, San Diego, CA, 1990) 176–233.
- [26] R.A. Howard, *Dynamic Probabilistic Systems, Vol II: Semi-Markov and Decision Processes* (John Wiley and Sons, New York, 1971).
- [27] R. Huslende, A combined evaluation of performance and reliability for degradable systems, in: *Proc. 1981 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (1981) 157–164.
- [28] S.M.R. Islam and H.H. Ammar, Numerical solution of Markov reward models using Laguerre functions, in: *Proc. 1st International Workshop on the Numerical Solution of Markov Chains*, Raleigh, NC (1990) 665–666.
- [29] B.R. Iyer, L. Donatiello and P. Heidelberger, Analysis of performability for stochastic models of fault-tolerant systems, *IEEE Trans. Comput.* **35** (10) (1986) 902–907.
- [30] P. Kubat, Reliability analysis for integrated networks with application to burst switching, *IEEE Trans. Comm.* **34** (6) (1986) 564–568.
- [31] V.G. Kulkarni, V.F. Nicola, R.M. Smith and K.S. Trivedi, Numerical evaluation of performability and job completion time in repairable fault-tolerant systems, in: *Proc. 16th International Symposium on Fault-Tolerant Computing* (IEEE Computer Society Press, Los Alamos, CA, 1986) 252–257.
- [32] V.G. Kulkarni, V.F. Nicola and K.S. Trivedi, The completion time of a job on multimode systems, *Adv. Appl. Probab.* **19** (1987) 932–954.
- [33] V.G. Kulkarni, V.F. Nicola and K.S. Trivedi, Effects of checkpointing and queueing on program performance, *Stochastic Models* **6** (4) (1990) 615–648.
- [34] J.C. Laprie, Dependable computing and fault-tolerance: Concepts and terminology, in: *Proc. 15th International Symposium on Fault-Tolerant Computing* (IEEE Computer Society Press, Los Alamos, CA, 1985) 2–7.
- [35] Y. Levy and P.E. Wirth, A unifying approach to performance and reliability objectives, in: M. Bonatti, ed., *Teletraffic Science for New Cost-Effective Systems, Networks and Services, ITC-12* (Elsevier Science Publishers, Amsterdam, 1989) 1173–1179.
- [36] M. Malhotra and K.S. Trivedi, Higher order methods for the transient analysis of Markov chains, in: *Proc. International Conference on the Performance of Distributed Systems and Integrated Communication Networks*, Kyoto, Japan, 1991.
- [37] J.F. Meyer, On evaluating the performability of degradable computing systems, *IEEE Trans. Comput.* **29** (8) (1980) 720–731.
- [38] J.F. Meyer, Closed-form solutions of performability, *IEEE Trans. Comput.* **31** (7) (1982) 648–657.
- [39] J.F. Meyer, A. Movaghar and W.H. Sanders, Stochastic activity networks: structure, behavior, and application, in: *Proc. International Workshop on Timed Petri Nets* (IEEE Computer Society Press, Los Alamos, CA, 1985) 106–115.
- [40] I. Mitrani and B. Avi-Itzhak, A many server queue with service interruptions, *Oper. Res.* **16** (1968) 628–638.
- [41] J.K. Muppala and K.S. Trivedi, Composite performance and availability analysis using a hierarchy of stochastic reward nets, in: G. Balbo, ed., *Proc. 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy (1991) 322–336.
- [42] J.K. Muppala and K.S. Trivedi, Numerical transient analysis of finite Markovian queueing systems, in: U.N. Bhat, ed., *Queueing and Related Models* (1991, to appear).
- [43] J.K. Muppala, S.P. Woollet and K.S. Trivedi, Real-time systems performance in the presence of failures, *IEEE Comput.* **24** (5) (1991) 37–47.
- [44] W. Najjar and J.L. Gaudiot, Reliability and performance modelling of hypercube-based multiprocessors, in: G. Iazeolla, P.J. Courtois and O.J. Boxma, Eds., *Computer Performance and Reliability* (North-Holland, Amsterdam, The Netherlands, 1988) 305–320.
- [45] V.F. Nicola, V.G. Kulkarni and K.S. Trivedi, Queueing analysis of fault-tolerant computer systems, *IEEE Trans. Software Eng.* **13** (3) (1987) 363–375.
- [46] K.R. Pattipati, Y. Li and H.A.P. Blom, A unified framework for the performability evaluation of fault-tolerant computer systems, in: *Proc. IEEE International Conference on System, Man, and Cybernetics*, Cambridge, MA (1989) 376–382.
- [47] A. Reibman, R. Smith and K.S. Trivedi, Markov and Markov reward model transient analysis: an overview of numerical approaches. *European J. Oper. Res.* **40** (1989) 257–267.
- [48] A.L. Reibman, Modeling the effect of reliability on performance, *IEEE Trans. Reliabil.* **39** (3) (1990) 314–320.
- [49] A.L. Reibman and K.S. Trivedi, Numerical transient analysis of Markov models, *Comput. Oper. Res.* **15** (1) (1985) 19–36.

- [50] A.L. Reibman and K.S. Trivedi, Transient analysis of cumulative measures of Markov model behavior, *Stochastic Models* 5 (4) (1989) 683–710.
- [51] W.H. Sanders, and J.F. Meyer, METASAN: A performance evaluation tool based on stochastic activity networks, in: *Proc. ACM-IEEE Comp. Soc. Fall Joint Comput. Conf.*, 1986.
- [52] W.H. Sanders and J.F. Meyer, Reduced base model construction for stochastic activity networks. *IEEE J. Selected Areas Comm.* 9 (1) (1991) 25–36.
- [53] R.M. Smith, K.S. Trivedi and A.V. Ramesh, Performability analysis: measures, an algorithm, and a case study, *IEEE Trans. Comput.* 37 (4) (1988) 406–417.
- [54] W. Stewart and A. Goyal, Matrix methods in large dependability models, Technical Report RC-11485, IBM T.J. Watson Res. Center, Yorktown Heights, NY 10598, 1985.
- [55] W.J. Stewart, MARCA: Markov chain analyzer, in: *Proc. 1st International Conference on the Numerical Solution of Markov Chains*, Raleigh, NC (1990) 30–52.
- [56] H. Tardif, K.S. Trivedi and A.V. Ramesh, Closed-form transient analysis of Markov chains, Technical Report CS-1988, Dept. of Computer Science, Duke University, Durham, NC 27706, 1988.
- [57] K.S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications* (Prentice-Hall, Englewood Cliffs, NJ, USA, 1982).
- [58] K.S. Trivedi, A.S. Sathaye, O.C. Ibe and R.C. Howe, Should I add a processor?, in: *Proc. 23rd Annual Hawaii International Conference on System Sciences* (IEEE Computer Society Press, Los Alamos, CA, 1990) 214–221.
- [59] L.T. Wu, Operational models for the evaluation of degradable computing systems, in: *Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (1982) 179–185.