Discrete and Continuous Models for the Performance of Reconfigurable Multistage Systems

Israel Koren, *Fellow, IEEE*, and Zahava Koren

Abstract—**When designing multistage multiprocessors consisting of a large number of components (e.g., processors, memories, and interconnection switches) we must expect some of the system components to become faulty while the system is in operation. In many cases, it may be desirable that the system be reconfigured into a fully-connected network of smaller dimensions, and its operation continued with some degradation in performance. However, before making a decision whether to support such graceful degradation we should estimate the expected reduction in system performance.**

In this paper, we analyze the performance of multiprocessor systems with a multistage interconnection network in the presence of faulty components. Models for estimating the system performance, as measured by its bandwidth and processing power, are developed for two different modes of operation. In the first mode, the operation of the system is fully synchronized and all processors which require memory access issue their requests simultaneously. In the second, each processor is allowed to issue its request at any time instant.

For each of the two modes of operation, two models are presented providing lower and upper estimates for the bandwidth of multistage systems and an upper estimate for their processing power. The expected degradation in the performance of the system predicted by these two models is then compared to simulation results.

Index Terms—**Bandwidth, faulty components, graceful degradation, multistage networks, processing power.**

I. Introduction

ADVANCES in VLSI technology and development of new computer-aided design tools enable the design and implementation of multiprocessing systems consisting of hundreds or even thousands of components. One important class of these multiprocessing systems includes the shared-memory multiprocessors where all processors can access a set of memory modules through a circuit-switching multistage interconnection network (MIN).

When implementing a complex multiprocessor, some of its components (like processors, memory modules, or interconnection links) should be expected to become faulty. In many cases the faulty components cannot be immediately repaired or replaced, yet the remaining units can be reconfigured into a functioning fully-connected (i.e., each processor is capable of communicating with every memory) network of smaller dimensions. Using the reconfigured system at a degraded rate of performance until a repair and/or replacement takes place can be beneficial, for example in a real-time computing system where even a relatively short down-time period may be intolerable.

The decision whether to support such graceful degradation of the system should depend upon its expected performance in the presence of faults. This is especially important in the case of multiprocessors with MIN's which provide a unique path between any processor and any memory module. These systems are inherently very sensitive to failures of any kind, since a single fault in any internal switch or link will render some memories unreachable from certain processors.

In this paper, we analyze the performance (over time) of reconfigurable multistage multiprocessors in the presence of faults. A commonly used measure for the performance of an interconnection network is its *bandwidth*. The bandwidth $BW(t)$ is defined as the expected number, at time t, of requests for the shared memory which are accepted per time unit. The bandwidth measures the effect of blocking which results either from memory conflicts (i.e., two or more requests directed to the same memory), from the sharing of paths by two or more processor–memory pairs (even when the memories involved are distinct), or, as in our case, from the presence of some faulty components. Another measure for system performance that we employ is the *processing power* defined as the average number of nonfaulty processors which are computing, i.e., operational processors which are neither communicating with the memory nor waiting for such a communication to be established. The processing power at time t is denoted by $C(t)$.

Two different types of models for analyzing the performance of multistage networks can be developed. The first one includes discrete models which assume a fully synchronized mode of operation. Here, time is divided into *network cycles* of fixed length, which equals the memory access time plus the network delay (twice the propagation delay of a signal through the network). Requests for memory access are issued at the beginning of a network cycle and all successful communications terminate at the end of the same cycle. The second type of model includes continuous models which assume an asynchronous mode of operation, i.e., each processor can issue a memory request at any time instant and the communication period can last an arbitrary length of time.

Two discrete models are presented in Section III generalizing previously suggested models ([12] and [13]) to allow the presence of faulty links, faulty processors, and faulty

Manuscript received February 20, 1990; revised August 13, 1990. This work was supported in part by NSF under Contract MIP-8805586. A preliminary version of this paper was presented at the 22nd HICSS, January 1989.

I. Koren is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.

Z. Koren is with the Department of Industrial Engineering and Operations Research, University of Massachusetts, Amherst, MA 01003.

IEEE Log Number 9101689.

memories. The first model is computationally simple, but too pessimistic for probabilities of request which are not close to 1. It assumes that a memory request blocked by the network is lost, an assumption that could result in a loss of bandwidth when the traffic is not very heavy. For very high request probabilities, this model coincides with the second model, which assumes that any processor whose memory request is blocked, reissues its request in the consecutive network cycle. This model is computationally more complex than the first one and a simplifying assumption must be made to make it mathematically tractable. Consequently, this model provides only an upper estimate for the network bandwidth. These two discrete models are compared to simulation results in Section IV.

In Section V, two similar models for the asynchronous mode of operation are presented based on the assumption that a blocked request is either lost or reissued, respectively. They too provide, for request rates which are not too high, lower and upper estimates for the bandwidth of the system but are computationally less complex than their discrete counterparts. The two continuous models are compared to simulation results in Section VI. Final conclusions are presented in Section VII.

II. Preliminaries

Consider a multistage circuit-switching interconnection network constructed of 2×2 switches which connect N processors (where $N = 2^k$) to N memories. Analysis of this kind of network can be generalized to the case where the number of processors is not necessarily a power of 2, the number of memories is different from the number of processors and finally, the network is built of $a \times b$ switches (see for example [12]). For the sake of clarity and brevity, however, we restrict our discussion here to the above mentioned simpler case.

The performance of unbuffered MIN's has been previously analyzed. In [3], [9], [12], and [13] it has been assumed that the network is fault-free. Networks in which faults may occur are considered in [2], [4], [10], and [11]. However, it is very difficult to extend the analysis in [2] to large values of N, while [11] assumes independence among the processor to memory paths, an unrealistic assumption which is omitted in our analysis. The analysis in [4] concentrates mainly on the number of operational paths in the network. The most recent work in [10] analyzes a special fault-tolerant MIN, called augmented shuffle-exchange network, and is based on two previously used simplifying assumptions, namely, all blocked requests are discarded and, arrivals of requests at different links in the MIN are statistically independent.

Although we are attempting to estimate the performance of a reconfigurable system, our analysis does not deal directly with the size of the reconfigured network, since calculating the exact probability of a fully-connected network of given dimensions is a problem with exponential complexity. Instead, our analysis takes into account all processors which are connected to at least one fault-free memory (termed *accessible* in [8]). We showed in [8] that due to the high correlation among the processor to memory paths in the network, the number of accessible processors can be viewed as a very close estimate of the number of processors in the fully-connected reconfigured network. This has been confirmed by our simulation results (reported in Sections IV and VI) in which actual reconfiguration took place.

The $N \times N$ interconnection network consists of $k = \log N$ stages, each containing $N/2$ switches as illustrated in Fig. 1. We assign numbers to the k stages in a descending order so that stage 0 is the last stage and its output links are connected to the memories, stage $(k-1)$ is the first stage and its input links are connected to the processors (see Fig. 1).

Let t be some given time instant, let $q_r(t)$ denote the probability that a processor is faulty at time t and let $p_r(t) = 1 - q_r(t)$ denote the probability that the processor is fault-free at time t. The functional form of $q_r(t)$ depends upon the statistical model assumed for the faults occurring in the network. The widely used model is the Poisson model, according to which the probability of a fault-free processor at time t is

$$p_r(t) = e^{-\lambda_r t} \tag{2.1}$$

where the failure rate λ_r is the average number of faults occurring in a processor per time unit.

Similarly, we denote by $q_m(t)$ $(p_m(t))$ the probability of a faulty (fault-free) memory and by $q_l(t)$ $(p_l(t))$ the probability of a faulty (fault-free) link, all at time t. Our fault model for the interconnection network is the link fault model [1], i.e., p_l is the probability that the wires connecting an output port of a switch and an input port of another switch are fault-free and the transmit/receive circuits on both ends are fault-free. We allow multiple link failures and consequently, entire switch faults are covered as well (a switch failure can be viewed as a failure of two links, thus having a probability of q_l^2). This is in line with a commonly made assumption that faults internal to circuits (switches in this case) which are implemented as integrated circuits are less likely to occur compared to faults in links connecting several IC's.

When Poisson distribution is assumed for fault occurrences, expressions similar to (2.1) are obtained for $p_m(t)$ and $p_l(t)$, with failure rates λ_m and λ_l, respectively. Although we use the Poisson model for the numerical examples, our analysis is general in the sense that it applies to any other statistical fault process, including models where the different components (namely, processors, links, and memories) follow different distribution laws and even models that allow repair of faulty components. The only requirement is that the model allows the calculation of the probabilities $p_r(t)$, $p_m(t)$, and $p_l(t)$.

For the purpose of our analysis we assume that the mean time between component failures is very large compared to the average length of the communication period (for both modes of operation, the synchronous and the asynchronous one). This implies that the status of the system components (i.e., faulty or fault-free) is constant for a large enough period of time allowing us to study the system's behavior under a statistical steady-state. We can, therefore, construct a Markovian process based on which the bandwidth and the processing power (for a given time instant t) will be calculated. For the synchronous mode of operation we construct a discrete Markov chain, while for the asynchronous mode we obtain a continuous Markov process. In both cases we view, for the purpose of the analysis,

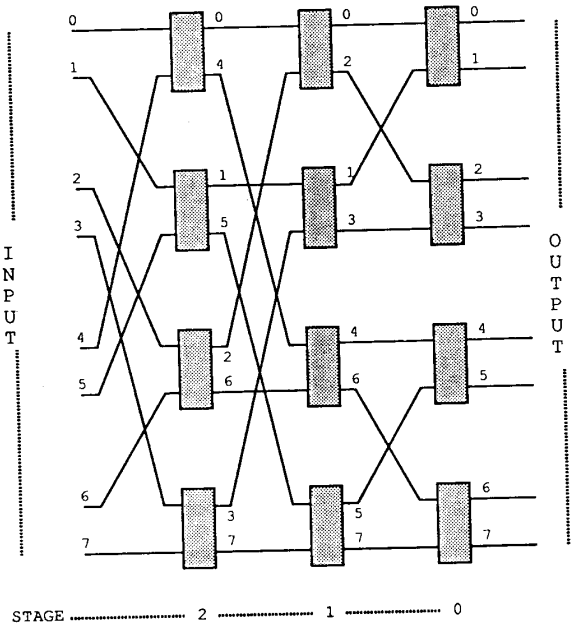


Fig. 1. An 8 × 8 multistage interconnection network.

the probabilities that the system components are fault-free as constants (for a fixed time instant t) and use for simplicity the notation p_r, p_m, and p_l. Similarly, we omit t and denote the bandwidth and processing power by BW and C, respectively.

III. Discrete Models

In this section, we present two discrete models for studying the degradation over time of the performance of a MIN in the presence of faults. Both models assume a fully synchronous mode of operation, i.e., processors may issue a memory request only at the beginning of a network cycle and the duration of every communication period between processor and memory is exactly one network cycle. The first model further assumes that blocked requests are discarded while the second one assumes that processors reissue their blocked requests in the next network cycle. We call these two models *nonpersistent* and *persistent*, respectively. The analysis of these two models appears in [6] and [7], respectively, and is briefly summarized here for the sake of completeness.

The Nonpersistent Model: The nonpersistent model generalizes a similar model [12] where the bandwidth has been calculated for a fault-free interconnection network.

Since in the nonpersistent model any blocked request is discarded, the system (observed at the beginning of network cycles) can be described by a memoryless stochastic process. Adopting the common assumption that the destinations of the memory requests are independent and uniformly distributed among the N memories, the network bandwidth can be obtained by multiplying the number of memories N by the probability that a given memory module is fault-free and has a request at its input. This last probability can be calculated iteratively, following a path leading to this memory, i.e., the probability of a request on an output link of a switch is calculated from the probability that such a request has been accepted at the input links to the same switch. The probability that a processor generates a request is denoted by p_a.

Let $X^{(i)}$ denote the event that there is a request for the memory on a particular output link of a switch in stage i (see Fig. 1). The probability of this event, denoted by $P\{X^{(i)}\}$, was shown in [6] to satisfy the following recurrence relation

$$P\left\{X^{(i)}\right\} = p_l \cdot P\left\{X^{(i+1)}\right\} - \frac{1}{4} p_l^2 \cdot \left(P\left\{X^{(i+1)}\right\}\right)^2$$

$$= 1 - \left(1 - \frac{1}{2} p_l \cdot P\left\{X^{(i+1)}\right\}\right)^2. \tag{3.1}$$

This recursion reduces to the one presented in [12] if fault-free (i.e., $p_l = 1$) 2 × 2 switches are assumed.

Equation (3.1) enables us to calculate the successive probabilities $P\{X^{(i)}\}$, starting from the processors outputs for which we have

$$P\left\{X^{(k)}\right\} = p_a p_r \tag{3.2}$$

up to the memory inputs yielding $P\{X^{(0)}\}$. Finally, to calculate the bandwidth for the nonpersistent model, denoted by $\mathrm{BW}^{\{np\}}$, we note that the memory and its input link can be faulty as well, hence

$$\mathrm{BW}^{\{np\}} = N \cdot P\left\{X^{(0)}\right\} \cdot p_m p_l. \tag{3.3}$$

Under the assumptions of the nonpersistent model, each fault-free processor whose memory access request is blocked, discards its request and resumes computing. The only processors which are not computing are therefore either faulty or successfully communicating with the memory. This assumption results in the following equation for calculating the processing power

$$C^{\{np\}} = N \cdot p_r - \mathrm{BW}^{\{np\}}. \tag{3.4}$$

This equation implies that a lower bandwidth results in a higher processing power. The nonpersistent model is thus, overly optimistic with regard to the processing power (as will be illustrated in Section IV) and should not be used for estimating it.

The persistent model, to be presented next, is more realistic than the previous one and provides an upper estimate for the bandwidth and a reasonably accurate estimate for the processing power.

The Persistent Model: In the persistent model, a blocked request is not discarded. Instead, the corresponding processor reissues its request in the next network cycle. This model generalizes a similar model [13] which was restricted to the case of fault-free components. The stochastic process required for describing the system behavior in the persistent model is more complex than that for the nonpersistent one. An exact state description should include the status of each processor, each link, and each memory, resulting in a prohibitively large

number of multidimensional states. We chose as an approximation for the state of the system a one-dimensional random variable Z, denoting the number of processors which are operational yet idle. These are the processors which are fault-free yet not computing because they are either communicating with some memory or waiting for such a communication to be established. This choice, as opposed to approximations in which the state of a single processor is considered, captures the dependence among the processors which is one of the main characteristics of an MIN.

The random variable Z is observed at the beginning of each network cycle, i.e., $Z(n)$ $(n = 0, 1, 2, \cdots)$ is the number of idle processors at the beginning of cycle n. In order for $Z(n)$ to be a Markov chain, we must add the following assumption: At the beginning of each cycle, all the existing requests (including both the new requests and the reissued ones) are randomly redistributed among all memories, regardless of their original destination. This "independence assumption" is made in [11] and in [13], and was shown to be inaccurate by Dias *et al.* [5]. Still, models based on this assumption can be utilized to obtain an upper estimate for the system's bandwidth, since it clearly tends to render the calculated bandwidth slightly higher than its actual value. In what follows, we show how $Z(n)$ can be used to estimate the network's bandwidth and processing power.

Denote by R the number, at time t, of operational processors $(0 \leq R \leq N)$ and thus, $N - R$ is the number of faulty processors. For a given value of R, $Z(n)$ can assume the values $0, 1, \cdots, R$. We denote by $P^{(R)}$ the one-step transition probability matrix whose (i, j)th element is

$$P^{(R)}_{i,j} = P\{Z(n+1) = j | Z(n) = i\}.$$

This is the probability that j processors request memory access at the beginning of cycle $n + 1$, given that i processors requested it at the beginning of cycle n. $Z(n) = i$ means that i processors are idle requesting memory access while $R - i$ are active performing internal processing. Out of the i requests, only d $(d \leq i)$ reach their destination and are accepted. The d corresponding processors become active again at the beginning of the $n + 1$ cycle, thus increasing the number of active processors to $R - i + d$. These $R - i + d$ active processors will generate g new requests, which will join the $i - d$ resubmitted ones. Consequently, $j = i + g - d$.

The transition from $Z(n) = i$ to $Z(n+1) = j$ involves two random variables: the number d of requests reaching their destination, and the number g of newly generated requests. The calculation of the transition probability matrix $P^{(R)}$ requires, therefore, the calculation of the following two probability matrices: D, whose (i, d) element is

$$D_{i,d} = P\{d \text{ requests accepted} | i \text{ requests submitted}\}$$

and G, whose (r, g) element is

$$G_{r,g} = P\{g \text{ requests generated} | r \text{ processors are active}\}.$$

Both matrices do not depend upon the actual value of R. Therefore, we define them as $(N+1) \times (N+1)$ matrices and use proper submatrices for any given value of R.

The calculation of $G_{r,g}$, the probability that r active processors will generate g requests, is straightforward. The r processors are independent, each having a probability p_a of generating a memory request, hence

$$G_{r,g} = \binom{r}{g} p_a^g (1 - p_a)^{r-g} \qquad r, g = 0, \cdots, N. \tag{3.5}$$

To calculate $D_{i,d}$, the probability that d out of i requests will reach their final destinations, we repeat k times the calculation for a single stage of switches. To this end, we denote by $\Phi_{u,v}$ the probability that v out of u requests at the inputs to the switches of any given stage will reach the inputs of the next stage. The elements $\Phi_{u,v}$ form an $(N+1) \times (N+1)$ transition probability matrix denoted by Φ. To find $\Phi_{u,v}$ we denote by a the number of switches in the observed stage which have one incoming request and by b the number of switches with two such requests. Clearly, $a + 2b = u$ and the probability of the pair (a, b) is

$$p\{a, b | u\} = \frac{\binom{\frac{N}{2}}{a} \cdot \binom{\frac{N}{2} - a}{b} \cdot 2^a}{\binom{N}{u}}. \tag{3.6}$$

Each of the a "single request" switches will propagate the incoming request depending on whether the appropriate link is fault-free or not. The probability of propagating the single request, denoted by α_1, is hence, $\alpha_1 = p_l$. The probability of not propagating the request, denoted by α_0, is $\alpha_0 = 1 - p_l$.

Denote by w the number of these "single request" switches propagating their incoming request and thus, $(a - w)$ single-request switches produce no output. Then,

$$P\{w | a\} = \binom{a}{w} \alpha_1^w \alpha_0^{a-w}. \tag{3.7}$$

Each of the b "double request" switches will propagate 2, 1, or 0 requests depending both on the destinations of the two incoming requests and on the status of the links (faulty or not). Denote by β_2, β_1, and β_0, the respective probabilities and by y, z, and $(b - y - z)$ the number of "double request" switches propagating 2, 1, and 0 requests, respectively, then,

$$P\{y, z | b\} = \binom{b}{y} \binom{b-y}{z} \beta_2^y \beta_1^z \beta_0^{b-y-z} \tag{3.8}$$

where $\beta_2 = 0.5 \cdot p_l^2$, $\beta_1 = p_l \cdot (1 - p_l) + 0.5 \cdot p_l$ and $\beta_0 = 1 - \beta_1 - \beta_2$.

Based on the probabilities in (3.6), (3.7), and (3.8) we obtain

$$\Phi_{u,v} = \sum_{\substack{a+2b=u \\ w+2y+z=v}} P\{a, b | u\} \cdot P\{w | a\} \cdot P\{y, z | b\}. \tag{3.9}$$

Having calculated Φ, the one-stage transition probability matrix, the k-stage transition probabilities can be obtained simply by raising Φ to the kth power. The (i, v) element of the resulting matrix $\Phi^k_{i,v}$ is the probability that v out of i original processors' requests will reach the memories. Finally, to calculate $D_{i,d}$ note that some of the v memories accessed may be faulty. The probability that out of v destinations, exactly d will be fault-free is

$$\binom{v}{d} p_m^d (1 - p_m)^{v-d} \tag{3.10}$$

and consequently,

$$D_{i,d} = \sum_{v=d}^{i} \Phi_{i,v}^{k} \cdot \binom{v}{d} p_m^d (1-p_m)^{v-d}. \tag{3.11}$$

The matrices G and D enable us to find $P^{(R)}$, the transition probability matrix of the Markov process Z, for every given number R of operational processors;

$$P_{i,j}^{(R)} = \sum_{j=i+g-d} D_{i,d} \cdot G_{R-i+d,g} \qquad i,j = 0,\cdots,R. \tag{3.12}$$

The matrix $P^{(R)}$ can now be used to calculate $\Pi^{(R)}$, the $(R+1)$-dimensional vector of the steady-state probabilities of the Markov chain, by solving the set of linear equations

$$\Pi^{(R)} \cdot P^{(R)} = \Pi^{(R)}; \qquad \sum_{i=0}^{R} \Pi_i^{(R)} = 1. \tag{3.13}$$

On the basis of the above steady-state probabilities the bandwidth $\mathrm{BW}^{\{p\}}$ and the processing power $C^{\{p\}}$ for the persistent model can be calculated. There is, however, no need to calculate both measures since they are functionally related in the currently presented model as shown in the following lemma.

Lemma: The bandwidth and processing power of a synchronous MIN for the persistent model satisfy

$$\mathrm{BW}^{\{p\}} = C^{\{p\}} \cdot \frac{p_a}{(1-p_a)} \qquad (\text{for } p_a < 1). \tag{3.14}$$

Proof: At the beginning of a network cycle, an average number of $C^{\{p\}}$ processors are active, joined by an average number of $\mathrm{BW}^{\{p\}}$ processors which have completed their memory access in the previous cycle and are now active too. Each of these $C^{\{p\}} + \mathrm{BW}^{\{p\}}$ processors has a probability p_a of issuing a request, hence an average of $(C^{\{p\}} + \mathrm{BW}^{\{p\}}) \cdot p_a$ requests are issued each cycle. Since the system must be in equilibrium, the expected number of generated requests per cycle must be equal to the expected number of accepted requests per cycle, $\mathrm{BW}^{\{p\}}$. Hence, $\mathrm{BW}^{\{p\}} = (C^{\{p\}} + \mathrm{BW}^{\{p\}}) \cdot p_a$ and (3.14) follows. ■

The conditional processing power for a given value of R, $C^{\{p\}}(R)$, can now be obtained using $\Pi^{(R)}$

$$C^{\{p\}}(R) = \sum_{i=0}^{R} (R-i) \cdot \Pi_i^{(R)}. \tag{3.15}$$

Averaging over all values of R using the appropriate probabilities yields

$$C^{\{p\}} = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot C^{\{p\}}(R). \tag{3.16}$$

$\mathrm{BW}^{\{p\}}$ can be obtained using (3.14). An alternative way of calculating $\mathrm{BW}^{\{p\}}$, which must be used if $p_a = 1$ (since (3.14) does not hold in this case), is the following. Define $\mathrm{BW}^{\{p\}}(R)$ as the conditional bandwidth for a given value of R. Then,

$$\mathrm{BW}^{\{p\}}(R) = \sum_{i=0}^{R} \Pi_i^{(R)} \sum_{d=0}^{i} d \cdot D_{i,d}. \tag{3.17}$$

And similarly to (3.16),

$$\mathrm{BW}^{\{p\}} = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot \mathrm{BW}^{\{p\}}(R). \tag{3.18}$$

IV. Simulation Results

In this section, we present some numerical comparisons between the two previously presented models and results of simulation runs. Our goal is to verify that, for the particular examples chosen, the nonpersistent and persistent models provide lower and upper estimates for the bandwidth, respectively, and to find out how close they are to the actual value (as obtained by simulation).

In the first step of the simulation, each processor, memory, and link has been set to be faulty or fault-free according to the probabilities p_r, p_m, and p_l, respectively. In the second step, the $N \times N$ network was reconfigured into a fault-free, fully-connected network of smaller dimensions which was run using the realistic protocol according to which a processor resubmits any blocked request to its original destination until it is fulfilled. The reconfiguration algorithm utilized is a "greedy" one. First, all faulty processors and memories are discarded from the system. Then, alternately, the processor connected to the smallest number of remaining memories and the memory connected to the smallest number of remaining processors are omitted, until a fault-free fully-connected system is obtained. This algorithm, which does not guarantee an "optimal" solution, provides good results for low fault probabilities which is the range we are mainly interested in. Both simulation steps were repeated many times, and the average bandwidth and processing power of the simulated systems were calculated. The results are depicted in Figs. 2–4 where the corresponding calculated 95% confidence intervals are indicated.

The results of the comparisons between the estimated and simulated values clearly depend upon the time t, the size of the network N, and the request probability p_a. Fig. 2 compares the values calculated for the bandwidth using the nonpersistent and persistent models to the value obtained through simulation, for a system with $N = 16$ processors and 16 memories. Time is measured in this figure in $1/\lambda_r$ units. The values chosen for the other failure rates are $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$. The bandwidth has been calculated as a function of time, for different values of p_a. The results for two values of p_a (i.e., $p_a = 0.2$ and $p_a = 0.6$) are shown in Fig. 2.

As is evident from Fig. 2, the results of the simulation lie between the two estimated values for the bandwidth, for "reasonable" values of t and p_a. For large values of t and p_a (p_a close to 1) the simulation results were lower than both estimated values. Large values of t and p_a are, however, impractical. Regarding t as the time since the last maintenance,

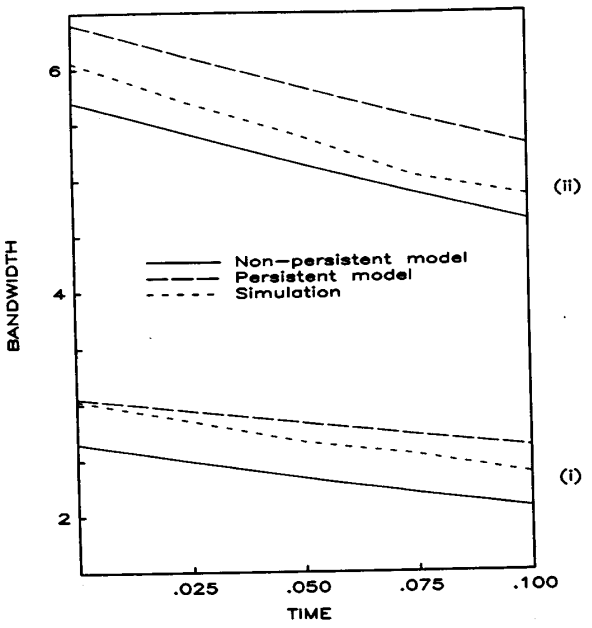


Fig. 2. Comparing the bandwidth BW(t) of a 16 × 16 synchronous network to its estimates using the persistent and nonpersistent models for (i) $p_a = 0.2$ and (ii) $p_a = 0.6$. 95% confidence intervals are less than: (i) ±0.04, (ii) ±0.065. (The failure rates are $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.)

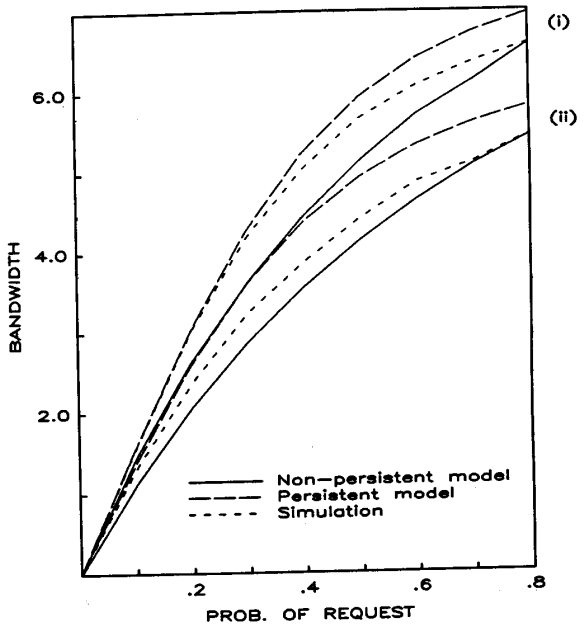


Fig. 3. The bandwidth of a 16 × 16 synchronous network as a function of p_a for (i) $t = 0$ and (ii) $t = 0.1$. 95% confidence intervals are less than: (i) ±0.01, (ii) ±0.07.

a large value of t indicates that the system is operating for long periods of time without any maintenance. On the other hand, a value of p_a which is close to 1 indicates that the processors access the memory very often and do very little internal processing.

Another important conclusion that can be drawn from Fig. 2 is that, for the example chosen (as well as for several other examples that we have run), the upper estimate for the bandwidth, calculated using the persistent model, is slightly closer to the simulation results than the lower estimate calculated using the simpler (nonpersistent) model.

Fig. 3 depicts the two estimates and the simulation results for the bandwidth, as a function of p_a, for two time instances ($t = 0$ and $t = 0.1$). As can be seen from Fig. 3, the upper estimate (using the persistent model) is closer to the simulation results for small values of p_a, while the lower estimate is closer for high values of p_a.

For high values of p_a the two estimates are getting relatively closer to each other; the loss of blocked requests is negligible when processors issue new requests at a high rate. Consequently, for high values of p_a, the computationally simpler model (the nonpersistent one) can be used for estimating the bandwidth.

We have also compared the estimated processing power of the synchronous system to its value obtained through simulation. Fig. 4 depicts the processing power of a 16 × 16 synchronous system as a function of time, for (i) $p_a = 0.2$ and (ii) $p_a = 0.4$. In case (i) we compare the simulation results to the results obtained from both models. As expected, the nonpersistent model is much too optimistic; the persistent model provides a better estimate for the processing power. We therefore omitted the nonpersistent model in case (ii) of Fig. 4. An important conclusion that can be drawn from Fig. 4 (and other cases that are not included here) is that the persistent model yields a reasonably good estimate for the processing power of the particular systems that we have analyzed.

Finally, Fig. 4 also demonstrates that, in this specific example, only a moderate reduction in performance is expected when the system is allowed to continue its operation in the presence of faults, making the support of graceful degradation a worthwhile endeavor.

V. Continuous Models

Most of the previously published analytic models for multistage networks assume synchronous operation of the system. An asynchronous system is analyzed in [2], but the analysis there is restricted to $N = 16$. In this section, we present two continuous models for studying the degradation over time in the performance of a multistage system under asynchronous operation, i.e., each processor can issue a memory request at any time, and the duration of the communication period between the processor and the memory is of arbitrary length. Similarly to the discrete models presented in Section III, the first continuous model is nonpersistent assuming that a blocked request is discarded, while the second one is a persistent model, in which a blocked request is reissued. These models provide lower and upper estimates for the bandwidth of the asynchronous system, and the persistent model can be used for estimating its processing power.

To study the behavior of the asynchronous system, a continuous stochastic process is required, and in order to make it

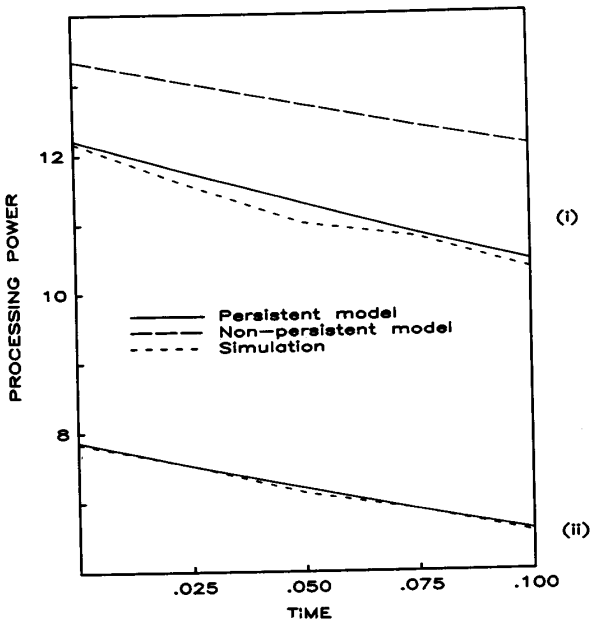


Fig. 4. The processing power of a 16×16 synchronous network as a function of time for (i) $p_a = 0.2$ and (ii) $p_a = 0.4$. 95% confidence intervals are less than: (i) ± 0.15, (ii) ± 0.1.

tractable, we approximate (as we have done in Section III) the state of the system by a one-dimensional random variable $Z(t)$. $Z(t)$ denotes the number, at time t, of operational yet idle processors, i.e., processors which are either communicating with some memory or waiting for such a communication to be established. Since fault rates are significantly smaller than the rate of memory requests, $Z(t)$ can reach a steady state for a fixed combination of faulty and nonfaulty components. We, therefore, omit t and denote by Z the state of the system. The steady-state probability distribution of Z still depends on t via the probabilities of faulty components.

For Z to be a birth and death Markovian process, the following assumptions have to be made. Each active (nonidle) processor generates a new request after a time which is exponentially distributed with a mean of $\frac{1}{\lambda}$. The duration of each successful communication is exponentially distributed with a mean of $\frac{1}{\mu}$. We also assume (as in Section III) that whenever a change in the state of the system occurs (whether it is a generation of a new request or the termination of a communication), all existing requests are randomly redistributed among all memories, regardless of their original destination.

As in Section III, the performance measures are first calculated for a given value of R which denotes the number (at time t) of operational processors. For a given R, Z can assume the values $0, 1, \cdots, R$, and $Z = i$ implies that i processors are requesting memory access while $R - i$ are computing. Let ρ_i denote the "birth" rate at state $Z = i$ (the rate of transition from i to $i + 1$), and let ν_i denote the "death" rate at state $Z = i$ (the rate of transition from i to $i - 1$). The conditional (on R) steady-state probabilities $\prod_i^{(R)}$ $(i = 0, \cdots, R)$ are obtained by solving the following set of equations

$$\Pi_i^{(R)} = \Pi_0^{(R)} \frac{\rho_0 \rho_1 \cdots \rho_{i-1}}{\nu_1 \nu_2 \cdots \nu_i} \qquad (i = 1, \cdots, R) \tag{5.1}$$

and

$$\sum_{i=0}^{R} \Pi_i^{(R)} = 1. \tag{5.2}$$

The preceding discussion applies to both continuous models. The two models, however, have different transition rates, which are calculated next.

The Nonpersistent Model: To find the transition rates for this model, note that at state $Z = i$, all i idle processors are actually communicating with some memory. Hence, the "death" rate for the nonpersistent model, denoted by $\nu_i^{\{np\}}$, is given by

$$\nu_i^{\{np\}} = i \cdot \mu. \tag{5.3}$$

For a transition from i to $i+1$ to take place, an active processor must generate a new request and this request must reach its destination (or it would be discarded). The "birth" rate for the nonpersistent model is therefore,

$$\rho_i^{\{np\}} = (R - i) \cdot \lambda \cdot \Gamma_i \tag{5.4}$$

where Γ_i is the probability that a new request will reach its destination, given that i processors are already communicating with i memories. The latter is equal to the probability that all links on the route to the destination and the requested memory are operational and not busy, given that i memories are busy. Combinatorial arguments yield

$$\Gamma_i = \left[\left(1 - 0.5 \frac{i}{(N-1)}\right) p_l\right]^k p_m. \tag{5.5}$$

The Persistent Model: For the persistent model, the rate $\rho_i^{\{p\}}$ at which a new request is generated at state $Z = i$ is

$$\rho_i^{\{p\}} = (R - i)\lambda. \tag{5.6}$$

The rate $\nu_i^{\{p\}}$ at which a communication ends, for $Z = i$, depends on the number d of requests out of i, which reach their destination. The probability matrix D, whose (i, d) element is

$$D_{i,d} = P\{d \text{ requests accepted} | i \text{ requests submitted}\},$$

has been calculated in Section III, (3.6)–(3.11). The "death" rate can be obtained from

$$\nu_i^{\{p\}} = \mu \sum_{d=0}^{i} d \cdot D_{i,d}. \tag{5.7}$$

The rest of the analysis is again common to both continuous models. The transition rates (from (5.3) and (5.4) for the nonpersistent model or from (5.6) and (5.7) for the persistent model) are now substituted into (5.1) to obtain the steady-state probabilities $\prod_i^{(R)}$; $(i = 0, \cdots, R)$. Based on these probabilities, the conditional processing power $C(R)$ and the conditional bandwidth $\mathrm{BW}(R)$ can be calculated. Since the

expressions for $C(R)$ and $\mathrm{BW}(R)$ are the same for both continuous models, the superscripts $\{p\}$ and $\{np\}$ are omitted.

Similarly to (3.15), the (conditional) processing power is given by

$$C(R) = \sum_{i=0}^{R} (R-i) \cdot \Pi_i^{(R)} \tag{5.8}$$

and thus, as in (3.16),

$$C = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1 - p_r p_l)^{N-R} \cdot C(R). \tag{5.9}$$

To obtain $\mathrm{BW}(R)$, note that at state $Z = i$ the rate of new arrivals is ρ_i [$\rho_i^{\{np\}}$ from (5.4) or $\rho_i^{\{p\}}$ from (5.6)]. Since the rate of new requests in the steady-state must be equal to the rate of communication completions, we have

$$\mathrm{BW}(R) = \sum_{i=0}^{R} \rho_i \cdot \Pi_i^{(R)} \tag{5.10}$$

and averaging over all values of R yields

$$\mathrm{BW} = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1 - p_r p_l)^{N-R} \cdot \mathrm{BW}(R). \tag{5.11}$$

VI. Numerical Results

In this section, we present some numerical comparisons between the two continuous models and results of simulation runs. Our first goal is to test whether, for the chosen examples, the nonpersistent and persistent models provide lower and upper estimates, respectively, for the bandwidth. In addition, we want to investigate how close these two estimates are to the simulation results, as a function of t and the ratio between the request rate λ and the "service" rate μ.

Similarly to the simulation for the synchronous model, components were set to be faulty according to their fault probabilities, the system was reconfigured to a smaller fully-connected one which was then run in an asynchronous manner. Estimates for the system bandwidth and processing power and the corresponding 95% confidence intervals have been calculated, and are reported in Figs. 5–7.

Fig. 5 compares the bandwidth calculated using the non-persistent and persistent models to that obtained through simulation, as a function of time, for a 16 × 16 system, and for two values of the memory request rate λ (i.e., $\lambda = 0.2$ and $\lambda = 0.5$). As before, time is measured in $1/\lambda_r$ units and the values chosen for the other failure rates are $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.

Fig. 6 depicts the bandwidth as a function of the memory request rate λ for two time instances ($t = 0$ and $t = 0.1$). As is evident from Figs. 5 and 6, the simulation results lie between the two estimated values for the bandwidth, for "reasonable" values of t and λ. For large values of t and λ the simulation results were lower than both estimated values. An important conclusion that can be drawn from these two figures is that, for this specific example, the upper estimate (obtained using the persistent model) is closer to the results of the simulation for small values of λ and t, while the lower estimate is closer for high values of λ and t. Similar behavior has been observed in other cases.

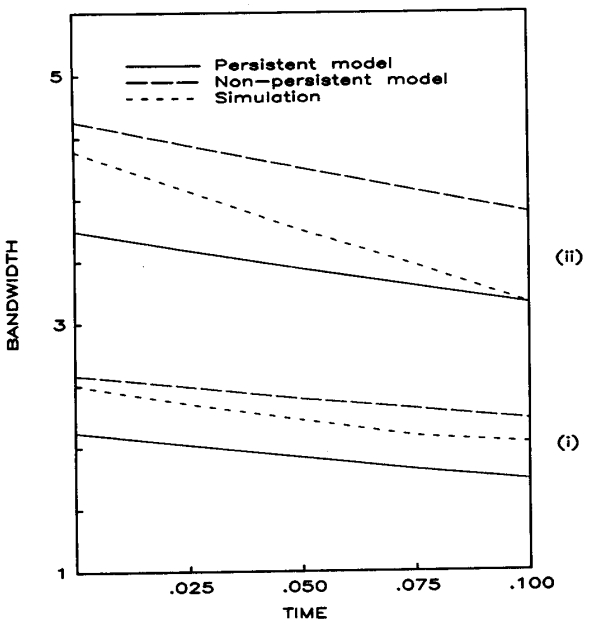


Fig. 5. Comparing the bandwidth $\mathrm{BW}(t)$ of a 16×16 asynchronous network to its estimates using the persistent and nonpersistent models for $\mu = 1$ and (i) $\lambda = 0.2$ or (ii) $\lambda = 0.5$. 95% confidence intervals are less than: (i) ±0.05, (ii) ±0.07. (The failure rates are $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.)

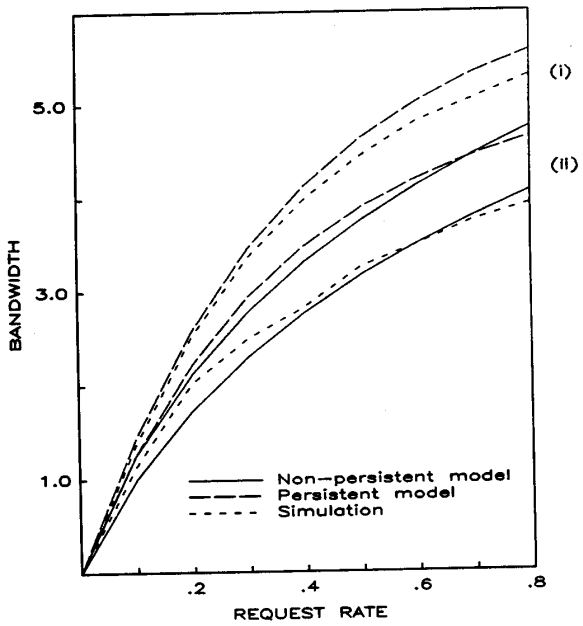


Fig. 6. The bandwidth of a 16 × 16 asynchronous network as a function of λ for $\mu = 1$ and (i) $t = 0$ or (ii) $t = 0.1$. 95% confidence intervals are less than: (i) ±0.02, (ii) ±0.07.

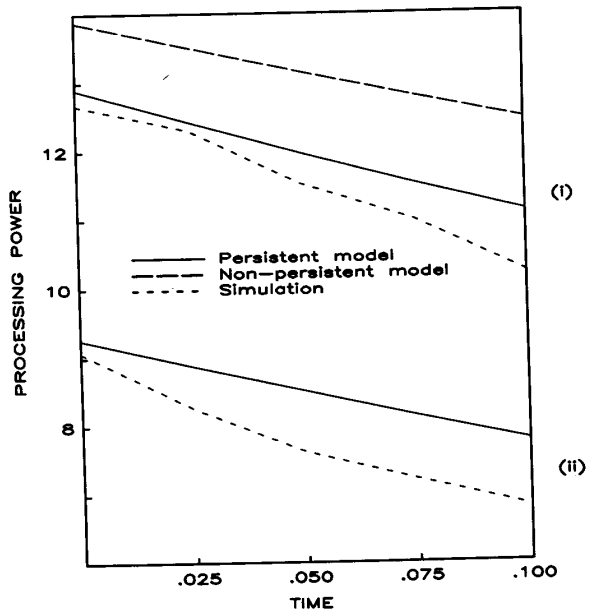


Fig. 7. The processing power of a 16 × 16 asynchronous network as a function of time for $\mu = 1$ and (i) $\lambda = 0.2$ or (ii) $\lambda = 0.5$. 95% confidence intervals are less than: (i) ± 0.22, (ii) ± 0.14.

Fig. 7 compares the estimated value of the processing power of a 16×16 asynchronous system to its value obtained through simulation, as a function of time, for two values of the request rate: (i) $\lambda = 0.2$ and (ii) $\lambda = 0.5$. As for synchronous systems, the nonpersistent model was found to be too optimistic; the persistent model provides, for all the analyzed examples, a better estimate for the processing power of the above system. The processing power obtained using the nonpersistent model is shown therefore, only for case (i) and is omitted in case (ii).

VII. Conclusions

The performance of multistage multiprocessor systems in the presence of faulty components has been analyzed in this paper. Two modes of operation have been studied, namely, the fully synchronous mode and the asynchronous mode. For each mode of operation we have developed two models allowing us to calculate the bandwidth and processing power of the multistage multiprocessor. The two models differ in the way blocked requests are treated. In the first (nonpersistent) model, blocked requests are discarded while in the second (persistent) model the processors reissue their blocked requests. The two discrete models presented in Section III generalize previous models in which a fault-free system has been assumed to be a system in which components may fail. The two continuous models presented in Section V are novel ones.

The operation of 16 × 16 synchronous and asynchronous reconfigurable systems has been simulated and the bandwidth and processing power have been calculated. These values were then compared to the estimated values using both the nonpersistent and the persistent models. For all simulated systems, the nonpersistent and persistent models provided lower and upper estimates for the bandwidth, respectively, for "practical" values of t and the request rate (or probability). Only the persistent model proved useful for estimating the processing power of a multistage system; it provided a better estimate than the nonpersistent model did.

References

[1] G. B. Adams, D. P. Agrawal, and H. J. Siegel, "A survey and comparison of fault-tolerant multistage interconnection networks," *IEEE Comput. Mag.*, vol. 20, pp. 14–27, June 1987.

[2] J. Arlat and J. C. Laprie, "Performance-related dependability evaluation of supercomputer systems," in *Proc. 13th Annu. Symp. Fault-Tolerant Comput.*, June 1983, pp. 276–283.

[3] L. N. Bhuyan and D. P. Agrawal, "Design and performance of generalized interconnection networks," *IEEE Trans. Comput.*, vol. C-32, pp. 1081–1090, Dec. 1983.

[4] V. Cherkassky and M. Malek, "Graceful degradation of multiprocessor systems," in *Proc. 1987 Int. Conf. Parallel Processing*, Aug. 1987, pp. 885–888.

[5] D. M. Dias and M. Kumar, "Comments on: Interference analysis of shuffle/exchange networks," *IEEE Trans. Comput.*, vol. C-31, pp. 546–547, June 1982.

[6] I. Koren and Z. Koren, "Analyzing the connectivity and bandwidth of multi-processors with multi-stage interconnection networks," in *Concurrent Computations: Algorithms, Architecture, and Technology*, S. K. Tewksbury, B. W. Dickinson and S. C. Schwartz, Eds. New York: Plenum, 1988, pp. 525–540.

[7] ——, "On the bandwidth of a multistage network in the presence of faulty components," in *Proc. 1988 Int. Conf. Distributed Comput. Syst.*, June 1988, pp. 26–32.

[8] ——, "On gracefully degrading multiprocessors with multistage interconnection networks," *IEEE Trans. Reliability*, vol. 38, no. 1, Special Issue on Reliability of Parallel and Distributed Computing Networks, pp. 82–89, Apr. 1989.

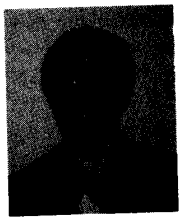
[9] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Comput.*, vol. C-32, pp. 1091–1098, Dec. 1983.

[10] V. P. Kumar and A. L. Reibman, "Failure dependent performance analysis of a fault-tolerant multistage interconnection network," *IEEE Trans. Comput.*, vol. 38, pp. 1703–1713, Dec. 1989.

[11] K. Padmanabhan and D. H. Lawrie, "Performance analysis of redundant-path networks for multiprocessor systems," *ACM Trans. Comput. Syst.*, vol. 3, no. 2, pp. 117–144, May 1985.

[12] J. H. Patel, "Performance of processor-memory interconnection for multiprocessors," *IEEE Trans. Comput.*, vol. C-30, pp. 771–780, Oct. 1981.

[13] S. Thanawastien and V. P. Nelson, "Interference analysis of shuffle/exchange networks," *IEEE Trans. Comput.*, vol. C-30, pp. 545–556, Aug. 1981.



Israel Koren (S'72–M'76–SM'87–F'91) received the B.Sc., M.Sc., and D.Sc., degrees from the Technion–Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in electrical engineering.

He is currently a Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. Previously he was with the Departments of Electrical Engineering and Computer Science at the Technion–Israel Institute of Technology. He also held visiting positions with the University of California, Berkeley, University of Southern California, Los Angeles, and University of California, Santa Barbara. He has been a consultant to Digital Equipment Corp., National Semiconductor, Tolerant Systems, and ELTA—Electronics Industries. His current research interests are fault-tolerant VLSI and WSI architectures, models for yield and performance, floor-planning of VLSI chips and computer arithmetic. He has edited and co-authored the book, *Defect and Fault-Tolerance in VLSI Systems, Vol. 1*, (New York: Plenum, 1989). He was also a Co-Guest Editor for IEEE TRANSACTIONS ON COMPUTERS, Special Issue on High Yield VLSI Systems, April 1989.

Zahava Koren received the B.A. and M.A. degrees in mathematics and statistics from the Hebrew University, Jerusalem, Israel, in 1967 and 1969, respectively, and the D.Sc. degree in operations research from the Technion–Israel Institute of Technology in 1976.

She is currently with the Department of Industrial Engineering and Operations Research at the University of Massachusetts, Amherst. Previously she has held positions with the Department of Statistics, University of Haifa, Departments of Industrial Engineering and Computer Science at the Technion, and the Department of Business and Economics, California State University in Los Angeles. Her main interests are stochastic analysis of computer networks, yield of integrated circuits, and reliability of computer systems.