DISCRETE AND CONTINUOUS MODELS FOR THE PERFORMANCE OF MULTI-STAGE SYSTEMS IN THE PRESENCE OF FAULTY COMPONENTS [1]

Israel Koren and *Zahava Koren*

Dept. of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

ABSTRACT

In this paper we analyze the performance of multi-processor systems with a multi-stage 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 need 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 multi-stage systems. The expected degradation in the performance of the system (in the presence of faulty components) predicted by these two models is then compared to simulation results.

1. 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.

When implementing a complex multiprocessor, some of its components (like processors, memory modules or interconnection switches) are expected to become faulty. In many cases the faulty components can not be immediately repaired or replaced, yet the remaining units can be reconfigured into a functioning network of smaller dimensions and a reduced rate of performance. An example might be a real-time computing system where even a relatively short down-time period may be intolerable. We would still like to use the system at the degraded rate of performance until a repair and/or replacement can take place.

[1]This work was supported in part by NSF under contract MIP-8805586.

The decision whether to support such a graceful degradation of the system should clearly depend upon its expected performance in the presence of faults. This is especially important in the case of multiprocessors with multistage interconnection networks 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 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* which is defined as the average number of non-faulty 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 models 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 for an arbitrary length of time.

Two discrete models are presented in Section 3 generalizing previously suggested models ([5] and [8]) to allow the presence of faulty links, faulty processors and faulty memories. The first model is computationally simple, but too pessimistic for low rates of request. It assumes that a memory request blocked by the network is lost, an assumption that for low traffic could result in a loss of bandwidth. For very high request rates, this first model coincides with the second model, which assumes that any processor whose memory request is blocked, re-issues its request in the consecutive network cycle. Although it seems more accurate than the first model, this second model is however, computationally more complex and provides only an upper estimate for the network bandwidth due to an additional assumption which will be explained later. These two discrete models are then compared to simulation results in Section 4.

In Section 5 two similar models for the asynchronous mode of operation are presented based on the assumption that a blocked request is either lost or re-issued, respectively. They too provide, for low request rates, 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 6. Final conclusions are presented in Section 7.

2. Preliminaries

Consider a multistage circuit-switching interconnection network constructed of 2×2 switches which connects N processors (where $N = 2^k$) to N memories.

The performance of unbuffered multistage interconnection networks has been previously analyzed. In [8], [5], [4] and [1] it has been assumed that the networks are fault-free. Faults are considered in [7] and [6], however, the analysis in [7] can not be extended to large values of N, while the analysis in [6] assumes independence among the processors, which is clearly unrealistic and is omitted in our analysis.

This $N \times N$ interconnection network consists of $k = logN$ 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.

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,

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

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. When Poisson distribution is assumed, similar expressions 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 applies for 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 probabilities $p_r(t)$, $p_m(t)$ and $p_l(t)$ can be calculated.

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 processing power and the bandwidth will be calculated. For the synchronous mode of operation we obtain a discrete Markov chain, while for the asynchronous mode we obtain a continuous Markov process. In both cases we will view, for the purpose of the analysis, the probabilities that the system components are fault-free as constants (for a fixed time instant t) and use for convenience the notation p_r, p_m and p_l.

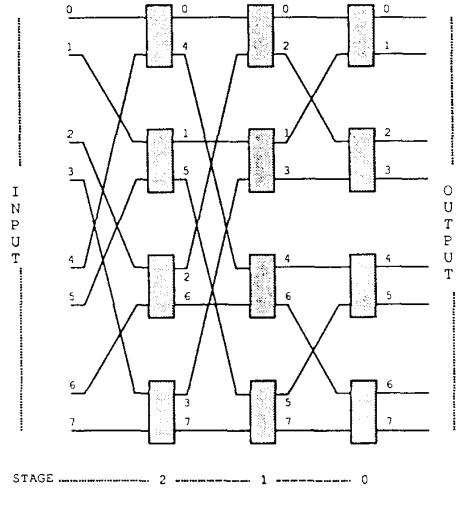

Figure 1: An 8×8 multi-stage interconnection network.

3. Discrete Models

In this section we present two discrete models for studying the degradation over time of the performance of a multistage interconnection network 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. The first model further assumes that blocked requests are discarded while the second one assumes that processors re-issue their blocked requests in the next network cycle. We call these two models *non-persistent* and *persistent*, respectively. The analysis of these two models appears in [2] and [3], and is summarized here for the sake of completeness.

The Non-Persistent Model: The non-persistent model generalizes a similar model [5] where the bandwidth has been calculated for a fault-free interconnection network.

Since in the non-persistent 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 has a request at its input. This last probability is 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 an output link of any switch in stage i (see Fig. 1). The probability of this event, denoted by $P\{X^{(i)}\}$, was shown in [2] to satisfy the following recurrence relation,

$$P\{X^{(i)}\} = p_l \cdot P\{X^{(i+1)}\} - \frac{1}{4}\ p_l^2 \cdot (P\{X^{(i+1)}\})^2 \qquad (3.1)$$

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

This recursion reduces to the one presented in [5] 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,

$$P\{X^{(k)}\} = p_a\ p_r \qquad (3.2)$$

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

$$BW^{\{np\}} = N \cdot P\{X^{(0)}\} \cdot p_m\ p_l \qquad (3.3)$$

The non-persistent model is, for low request rates, too pessimistic with regard to the bandwidth, providing only a lower estimate for this measure. However, since the processors, after having discarded their memory access requests, are free for computing, this model is overly optimistic with regard to the processing power as will be illustrated in Section 4. 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 re-issues its request in the next network cycle. This model generalizes a similar model [8] which was restricted to the case of fault-free components. The stochastic process required for describing the system's behavior in the persistent model is more complex than that for the non-persistent one. An exact description of the system's state should include the state of each processor, each link and each memory resulting in a prohibitively large number of multi-dimensional 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 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 a multistage interconnection network.

The random variable Z is observed at the beginning of each network cycle, i.e., $Z(n)$ (n=0,1,2,...) 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 re-issued ones) are randomly redistributed among all memories, regardless of their original destination. This "independence assumption" is made in [8] and in [6], and although it is not completely accurate (as demonstrated in [9]) it is necessary in order to make the analysis tractable. However, models based on this assumption can be utilized for obtaining 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 calculate the network's bandwidth and processing power.

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

$$P_{i,j}^{(R)} = P\{\ Z(n+1) = j \mid Z(n) = i\ \}$$

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 destinations 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 destinations, 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

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

and G, whose (r,g) element is

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

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 sub-matrices 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,

$$G_{r,g} = \binom{r}{g} {p_a}^g (1-p_a)^{r-g} \qquad r,g = 0,...,N \quad (3.4)$$

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

$$p\{a,b|u\} = \frac{\binom{\frac{N}{2}}{a} \cdot \binom{\frac{N}{2}-a}{b} \cdot 2^a}{\binom{N}{u}} \quad (3.5)$$

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, $(a-w)$ switches produce no output. Then,

$$P\{w|a\} = \binom{a}{w} {\alpha_1}^a (1-\alpha_1)^{a-w} \quad (3.6)$$

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,

$$P\{y,z|b\} = \binom{b}{y}\binom{b-y}{z} {\beta_2}^y {\beta_1}^z {\beta_0}^{b-y-z} \quad (3.7)$$

where,

$$\beta_2 = 0.5 \cdot p_l^2$$

$$\beta_1 = p_l \cdot (1-p_l) + 0.5 \cdot p_l$$

$$\beta_0 = 1 - \beta_1 - \beta_2$$

Based on the probabilities in (3.5), (3.6) and (3.7) we obtain,

$$\Phi_{u,v} = \sum_{\substack{a+2b=u \\ w+2y+z=v}} P\{a,b|u\} \cdot P\{w|a\} \cdot P\{y,z|b\} \quad (3.8)$$

Having calculated Φ, the one-stage transition probability matrix, the k-stage transition probabilities can be obtained simply by raising Φ to the k-th 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 memories, exactly d will be fault-free is,

$$\binom{v}{d} {p_m}^d (1-p_m)^{v-d} \quad (3.9)$$

and consequently,

$$D_{i,d} = \sum_{v=d}^{i} \Phi^k_{i,v} \cdot \binom{v}{d} {p_m}^d (1-p_m)^{v-d} \quad (3.10)$$

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,

$$P^{(R)}_{i,j} = \sum_{j=i+g-d} D_{i,d} \cdot G_{R-i+d,g} \qquad i,j = 0,...,R \quad (3.11)$$

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

$$\Pi^{(R)} \cdot P^{(R)} = \Pi^{(R)} \quad ; \quad \sum_{i=0}^{R} \Pi^{(R)}_i = 1 \quad (3.12)$$

On the basis of the above steady-state probabilities the bandwidth $BW^{(p)}$ 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 multistage interconnection network for the persistent model satisfy,

$$BW^{\{p\}} = C^{\{p\}} \cdot \frac{p_a}{(1-p_a)} \qquad (\text{for } p_a < 1) \qquad (3.13)$$

Proof: At the beginning of a network cycle, an average number of $C^{\{p\}}$ processors are active, joined by an average number of $BW^{\{p\}}$ processors which have completed their memory access in the previous cycle and are now active too. Each of these $C^{\{p\}} + BW^{\{p\}}$ processors has a probability p_a of issuing a request, hence an average of $(C^{\{p\}} + BW^{\{p\}}) \cdot p_a$ 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, $BW^{\{p\}}$. Hence, $BW^{\{p\}} = (C^{\{p\}} + BW^{\{p\}}) \cdot p_a$ and (3.13) follows. ∎

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

$$C^{\{p\}}(R) = \sum_{i=0}^{R} (R-i) \cdot \Pi_i^{(R)} \qquad (3.14)$$

Averaging over all values of R using the appropriate probabilities yields,

$$C^{\{p\}} = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot C^{\{p\}}(R) \qquad (3.15)$$

$BW^{\{p\}}$ can be obtained using (3.13). An alternative way of calculating $BW^{\{p\}}$, which must be used if $p_a = 1$ (since (3.13) does not hold in this case), is the following. Define $BW^{\{p\}}(R)$ as the conditional bandwidth for a given value of R. Then,

$$BW^{\{p\}}(R) = \sum_{i=0}^{R} \Pi_i^{(R)} \sum_{d=0}^{i} d \cdot D_{i,d} \qquad (3.16)$$

And similarly to (3.15),

$$BW^{\{p\}} = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot BW^{\{p\}}(R) \qquad (3.17)$$

4. Simulation Results

In this section we present some numerical comparisons between the two previously presented models and results of simulation runs. Our goal was to verify that the non-persistent and persistent models provide lower and upper estimates, respectively, for the bandwidth, and to find out how close they are to the actual value. These comparisons clearly depend upon the time t, the size of the network N and the request probability p_a.

Figure 2 compares the values calculated for the bandwidth using the non-persistent and persistent models to the value obtained through simulation, as a function of time, 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 and the results for two values of p_a (i.e., $p_a = 0.2$ and $p_a = 0.6$) are shown in Figure 2. In the simulation we have introduced a timeout mechanism which is required to avoid the situation where a processor re-issues indefinitely its request to access either a faulty memory or an inaccessible one (due to some link faults). Without such a timeout mechanism, the value obtained for the bandwidth through simulation decreases very rapidly with increasing time t.

As is evident from Figure 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, a large value of t indicates that the system is operating for large 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 Figure 2 is that the upper estimate for the bandwidth, calculated using the persistent model, is closer to the simulations results for relatively smaller values of t, while for larger values of t the lower estimate calculated using the simpler (non-persistent) model is closer. A similar phenomenon is observed in Figure 3 with regard to p_a. Figure 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 Figure 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 non-persistent one) can be used for estimating the bandwidth.

We have then compared the estimated processing power of the synchronous system to its value obtained through simulation. Figure 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 non-persistent model is too optimistic; the persistent model provides a better estimate for the processing power. We therefore omitted the non-persistent model in

case (ii). An important conclusion that can be drawn from Figure 4 is that the persistent model yields a reasonably good estimate for the processing power of the system.

Finally, Figure 4 also demonstrates the fact that 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 a graceful degradation a worthwhile endeavor.

5. Continuous Models

In this section we present two continuous models for studying the degradation over time of the performance of a multistage interconnection network in the presence of faults. In both models the operation of the network is asynchronous, 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 3, the first continuous model is non-persistent assuming that a blocked request is discarded, while the second one is a persistent model, in which a blocked request is re-issued. These models will provide lower and upper estimates for the bandwidth of the asynchronous system, and the persistent model will be used for estimating its processing power.

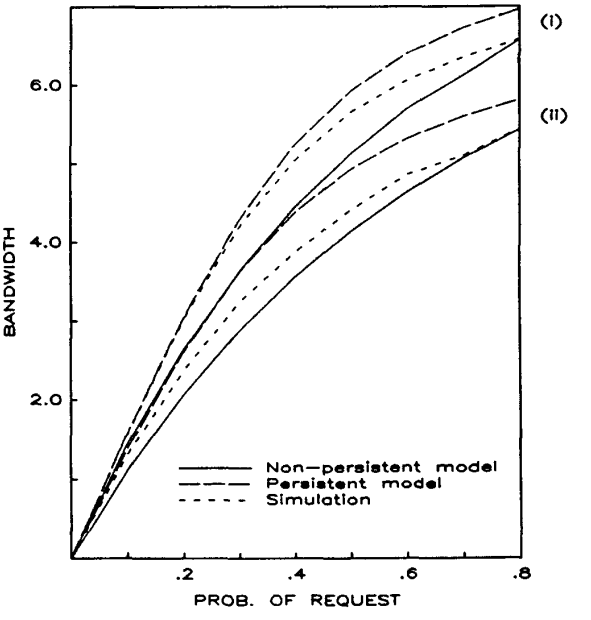

Figure 3: The bandwidth of a 16×16 synchronous network as a function of p_a for (i) $t = 0$ and (ii) $t = 0.1$.

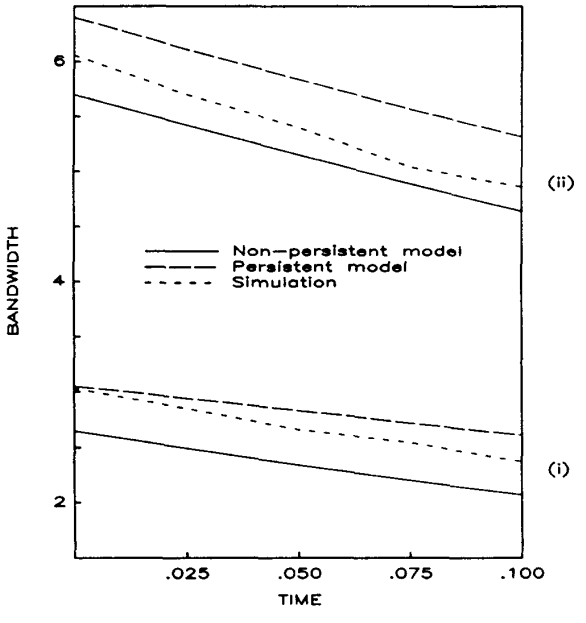

Figure 2: Comparing the bandwidth $BW(t)$ of a 16×16 synchronous network to its estimates using the persistent and non-persistent models for (i) $p_a = 0.2$ and (ii) $p_a = 0.6$. (Failure rates: $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.)

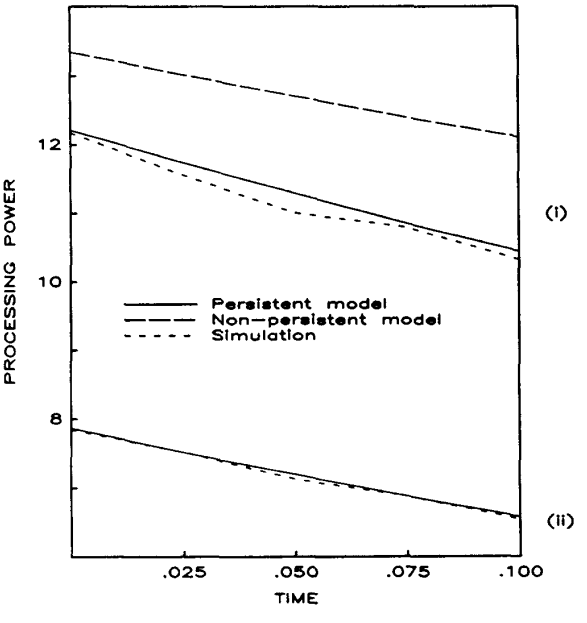

Figure 4: The processing power of a 16×16 synchronous network as a function of time for (i) $p_a = 0.2$ and (ii) $p_a = 0.4$.

The stochastic process which is needed to study the behavior of the system is a continuous one, and in order to make it tractable, we approximate (as we have done in Section 3) 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 non-faulty 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 (non-idle) 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 3) 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 3, the performance measures will first be 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 $0, 1, ..., R$, 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 $\Pi_i^{(R)}$ $(i = 0, ..., R)$ are obtained by solving the following set of equations,

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

and

$$\Pi_0^{(R)} + \Pi_1^{(R)} + ... + \Pi_R^{(R)} = 1 \tag{5.2}$$

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

The Non-Persistent 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 non-persistent 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 non-persistent 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

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

has been calculated in Section 3, equations (3.5) through (3.10). 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 non-persistent model or from (5.6) and (5.7) for the persistent model) are now substituted into (5.1) to obtain the steady-state probabilities $\Pi_i^{(R)}$; $(i = 0, ..., R)$. Based on these probabilities the conditional processing power $C(R)$ and the conditional bandwidth $BW(R)$ can be calculated. Since the expressions for $C(R)$ and $BW(R)$ are the same for both continuous models, the superscripts $\{p\}$ and $\{np\}$ are omitted.

Similarly to (3.14), 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 (3.15),

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

To obtain $BW(R)$, 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

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

Hence, as in (3.17),

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

6. 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 the non-persistent 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.

Figure 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 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 $\lambda_l/\lambda_r = 0.2$. We again had to introduce a timeout mechanism into the simulator to avoid the situation where a processor re-issues indefinitely its request to access either a faulty memory or an inaccessible one (due to some link faults).

Figure 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 Figures 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 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.

Figure 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 non-persistent model was found to be too optimistic; the persistent model provides a better estimate for the processing power. The processing power obtained using the non-persistent model is shown therefore, only for case (i) and is omitted in case (ii). An important conclusion that can be drawn from Figure 7 is that the estimated processing power calculated for the persistent model is close to the simulation results only for small values of λ and t.

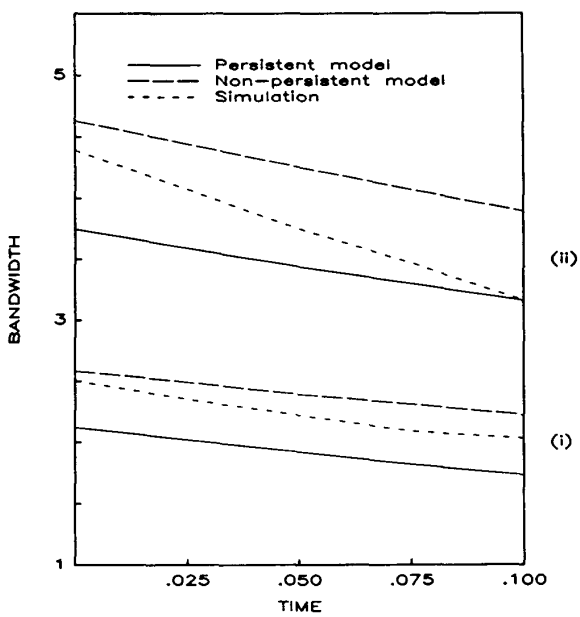

Figure 5: Comparing the bandwidth $BW(t)$ of a 16×16 asynchronous network to its estimates using the persistent and non-persistent models for $\mu = 1$ and (i) $\lambda = 0.2$ or (ii) $\lambda = 0.5$. (Failure rates: $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.)

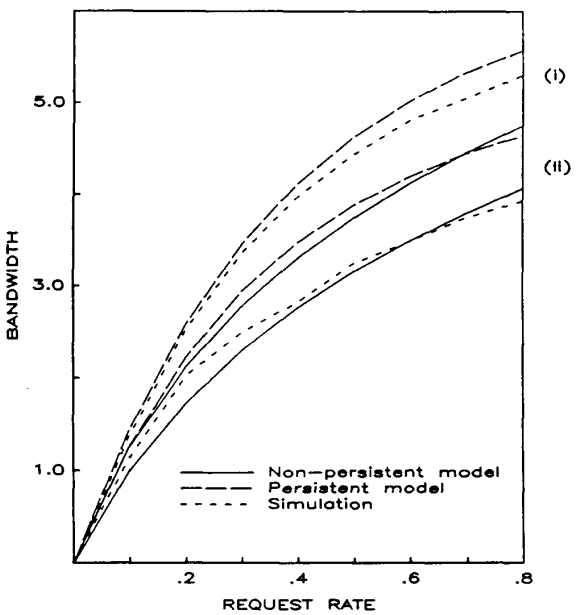

Figure 6: The bandwidth of a 16×16 asynchronous network as a function of λ for $\mu = 1$ and (i) $t = 0$ or (ii) $t = 0.1$.

7. 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 one these requests are discared while in the second, the corresponding processors re-issue their requests. The two models are therefore, called non-persistent and persistent model, respectively. The two discrete (non-persistent and persistent) models presented in Section 3 generalize previously presented models which have assumed fault-free components, while the two continuous models presented in Section 5 are novel ones.

The operation of 16×16 synchronous and asynchronous systems has been simulated and the bandwidth and processing power have been calculated. These values were than compared to the estimated values using the non-persistent and persistent models. For both modes of operation the non-persistent and persistent models provide lower and upper estimates for the bandwidth, respectively, for "practical" values of t and the request rate (or probability). Only the persistent model is useful when estimating the processing power of a multistage system; it provides a better (upper) estimate than the non-persistent model does.

To determine which set of models to use, the discrete models or the continuous models, one should take into account both the protocol employed by the interconnection network and the computational complexity of the models. The continuous models which are based on a birth and death Markov process are computationally simpler than the persistent discrete model which is based on a discrete Markov chain.

8. References

[1] L. N. Bhuyan and D. P. Agrawal, "Design and Performance of Generalized Interconnection Networks," *IEEE Trans. on Computers,* Vol. C-32, Dec. 1983, pp. 1081-1090.

[2] I. Koren and Z. Koren, "Analyzing the Connectivity and Bandwidth of Multi-processors with Multi-stage Interconnection Networks," Concurrent Computations: Algorithms, Architecture, and Technology, S.K. Tewksbury, B.W. Dickinson and S.C. Schwartz (eds.), Plenum, 1988, pp. 525-540.

[3] I. Koren and Z. Koren, "On the Bandwidth of a Multistage Network in the Presence of Faulty Components," *Proc. of the 1988 Internl. Conf. on Distributed Computing Systems,* June 1988, pp. 26-32.

[4] C. P. Kruskal and M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Trans. on Computers,* Vol. C-32, Dec. 1983, pp. 1091-1098.

[5] J. H. Patel, "Performance of Processor-Memory Interconnection for Multiprocessors," *IEEE Trans. on Computers,* Vol. C-30, Oct. 1981, pp. 771-780.

[6] K. Padmanabhan and D. H. Lawrie, "Performance Analysis of Redundant-Path Networks for Multiprocessor Systems," *ACM Trans. on Computer Systems,* Vol. 3, No. 2, May 1985, pp. 117-144.

[7] J. Arlat and J. C. Laprie, "Performance-related Dependability Evaluation of Supercomputer Systems," *Proc. of the 19-th Annual Symp. on Fault-Tolerant Computing,* June 1983, pp. 276-283.

[8] S. Thanawastien and V. P. Nelson, "Interference Analysis of Shuffle/Exchange Networks," *IEEE Trans. on Computers,* Vol. C-30, Aug. 1981, pp. 545-556.

[9] D. M. Dias and M. Kumar, "Comments on: Interference Analysis of Shuffle/Exchange Networks," *IEEE Trans. on Computers,* Vol. C-31, June 1982, pp. 546-547.

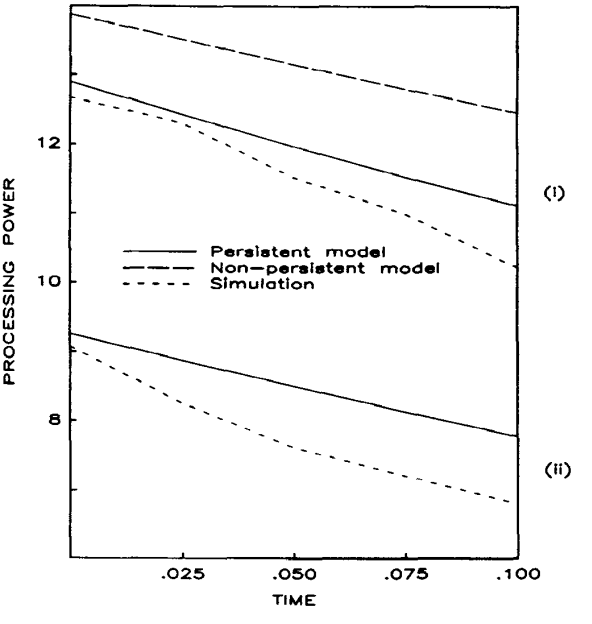

Figure 7: The processing power of a 16×16 asynchronous network as a function of time for $\mu = 1$ and (i) $\lambda = 0.2$ or (ii) $\lambda = 0.5$.