ON THE BANDWIDTH OF A MULTI-STAGE NETWORK IN THE PRESENCE OF FAULTY COMPONENTS[1]

Israel Koren[2] and *Zahava Koren*

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

ABSTRACT

Multi-processing systems consisting of a large number of components (e.g., processors, memory modules and interconnection switches) are now being designed and implemented. Improvements in technology have reduced the failure rates of all system components. However, the large increase in the number of components per system has more than offset the increase in reliability of a single component. Therefore, we must expect some of the hundreds (or even thousands) of system components to become faulty while the system is in operation.

In many cases, the continued operation of a system in the presence of faulty components, even with some degradation in performance, is desirable. The decision whether to support such graceful degradation should depend on the estimated reduction in system performance. In this paper we analyze the performance of multi-processor systems with a multi-stage interconnection network in the presence of faulty components. We present two models, providing lower and upper estimates for the bandwidth of these systems. We then compare the expected degradation in the performance of the multi-processor in the presence of faulty components for these two models through some numerical examples.

1. Introduction

Advances in VLSI technology and development of new computer-aided design tools, enable the design and implementation of multi-processing systems consisting of a very large number of components. One important class of these multiprocessing systems includes the shared-memory multi-processors where all processors can access a set of memory modules through an unbuffered multi-stage interconnection network.

When implementing a complex multi-processor, some of its components (like processors, memory modules or interconnection links) are expected to become faulty. In many cases the faulty components can not be immediately repaired or replaced and we would still like to use the system at a degraded rate of performance until a repair and/or replacement can take place. An example might be a real-time computing system where even a relatively short down-time period may be intolerable.

The decision whether to support 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 multi-processors with multi-stage interconnection networks which are inherently very sensitive to failures of any kind. These networks usually provide a unique path between any processor and any memory module and therefore, 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 multi-stage multi-processors in the presence of faults. A commonly used measure for the performance of an interconnection network is the *bandwidth*. The bandwidth $BW(t)$ is defined as the expected number, at time t, of requests for the shared memory which are accepted per cycle. The bandwidth measures the effect of blocking which results either from requests being 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.

We present two models for calculating the bandwidth of the multi-stage network. These models generalize previously suggested models ([4] and [5]) to allow the presence of faulty links, faulty processors and faulty memories. The first model is computationally simple but too pessimistic since it assumes that a memory request blocked by the network is lost. The second model assumes that any processor whose memory request is blocked, re-issues its request in the consecutive network cycle. This model which is computationally more complex, provides an upper estimate for the network bandwidth.

The second model also allows the calculation of other measures for the system's performance. For example, the *processing power*, 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 will be denoted by $C(t)$.

[1]Supported in part by NSF under contract DCR-85-09423.
[2]On leave from the Technion, Haifa 32000, Israel

The rest of the paper is organized as follows. In the next section we introduce several assumptions and notations. In Section 3 the first and simpler model is presented. The more complex model is introduced and analyzed in Section 4. These models are then compared through some numerical examples in Section 5. Final conclusions are presented in Section 6.

2. Preliminaries

Consider a multi-stage interconnection network constructed of 2×2 unbuffered switches which connects N processors (where $N = 2^k$) to N memories. Our analysis 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 [1]. For the sake of clarity and brevity however, we restrict our discussion here to the above mentioned simpler case.

An $N \times N$ interconnection network is constructed of $k = logN$ stages, each containing $N/2$ switches as illustrated in Fig. 1. The performance of multi-stage interconnection networks has been previously analyzed (e.g., [4], [5], [1] and [3]), but in most previous studies it has been assumed that either the networks are fault-free, or that faults can occur in the interconnection network only, while the processors and memories were assumed to be fault-free. In our analysis we allow processors and memories to become faulty in addition to faults occurring in the interconnection network. Moreover, we study the behavior of these systems over time.

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 $p_r(t)$ 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. 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 our 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.

We consider a system which operates in a synchronous circuit-switching fashion, i.e., time is divided into *network cycles* of equal length. Any request for memory access is issued at the beginning of a network cycle and all successful communications terminate at the end of the cycle. The length of a network cycle is equal to the memory access time plus the network delay (twice the propagation delay of a signal through the k stages). For analysis purposes we assume that the mean time between component failures is very large compared to the length of the network cycle, implying that once a component fails, the next failure will occur after a very long period of time. This enables us to study the system's behavior under a statistical steady-state, resulting from its having a fixed combination of faulty and non-faulty components for a relatively long time.

3. Model I

In this section we present a simplified model for studying the degradation over time of the bandwidth $BW(t)$ of a multi-stage interconnection network in the presence of faults. This model generalizes the analysis in [4] where the bandwidth of a fault-free interconnection network was calculated. The analysis has been generalized in [2] to an environment in which faults may occur and is briefly summarized here for the sake of completeness.

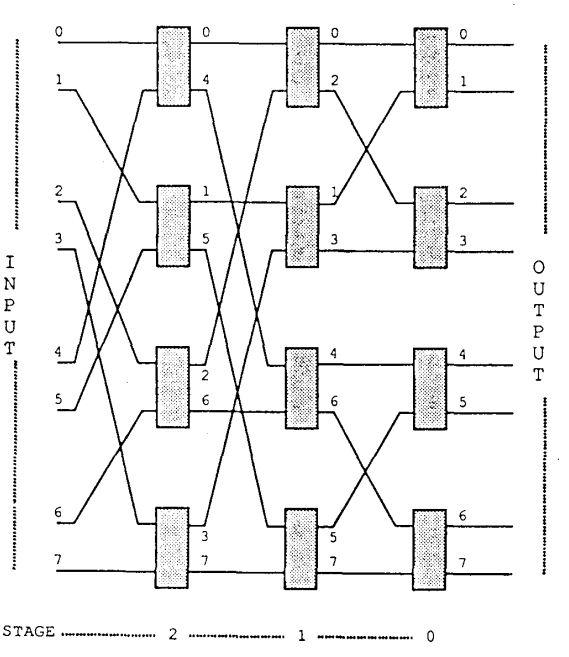

Figure 1: An 8×8 multi-stage interconnection network.

The time-dependency of $BW(t)$ is the result of the time-dependency of the probabilities that the system components are fault-free, i.e., $p_r(t)$, $p_l(t)$ and $p_m(t)$. Since the duration of a single network cycle is (according to our assumption) substantially smaller than the mean time between component failures, we can perform a steady-state analysis of the bandwidth. The system is observed at some arbitrary time instant t, which can be viewed as a constant throughout the analysis, and will therefore be omitted from the notations for the sake of simplicity. However, the reader should bear in mind that all the probabilities and the resulting bandwidth (BW) are functions of t.

We adopt here the common assumption that the destinations of the memory requests are independent and uniformly distributed among the N memories. Therefore, the network bandwidth can be obtained by multiplying the number of memories N by the probability that a given memory module is non-faulty and 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.

To simplify our discussion we say that a link is in state 1 (0) if it has (has not) a request for the memory. A faulty link is considered to be in state 0. The probability of a request on a link is thus the probability that this link is in state 1. We assign numbers to the $k = logN$ 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 one and its inputs are connected to the processors (see Fig. 1). Consider a switch in stage i and denote its outputs $X^{(i)}, Y^{(i)}$. Its input links are the outputs of (two different) switches in stage $(i+1)$ and are denoted by $X^{(i+1)}$ and $Y^{(i+1)}$. Based on our assumption that memory requests are uniformly distributed among the memories, the probability that an incoming request will be routed to any output link is the same. Hence, it is sufficient to consider only a single output link and derive an expression for the probability that it is at state 1, i.e., $P\{X^{(i)} = 1\}$.

Since a request for a memory module can reach the output link of a switch through any of the two input links, the state probability $P\{X^{(i)} = 1\}$ of the given output link has to be calculated from the joint probabilities of these input links, i.e.,

$$P\{(X^{(i+1)}, Y^{(i+1)}) = (u,v)\}; \quad u,v = 0,1.$$

This calculation is performed using transition probabilities which take into account the status (faulty or fault-free) of the (physical) input links and the destinations of the incoming requests. Since memory modules are assumed to be equivalent, the incoming requests are routed to any of the two output links with probability 0.5. Consequently, the transition probabilities between the two inputs and the output of a switch are,

$$\begin{aligned}
&P\{X^{(i)} = 1 \;/\; (X^{(i+1)}, Y^{(i+1)}) = (0,0)\} = 0 \\
&P\{X^{(i)} = 1 \;/\; (X^{(i+1)}, Y^{(i+1)}) = (0,1)\} = \tfrac{1}{2}\, p_l \\
&P\{X^{(i)} = 1 \;/\; (X^{(i+1)}, Y^{(i+1)}) = (1,0)\} = \tfrac{1}{2}\, p_l \\
&P\{X^{(i)} = 1 \;/\; (X^{(i+1)}, Y^{(i+1)}) = (1,1)\} = p_l - \tfrac{1}{4}\, p_l^2
\end{aligned} \tag{3.1}$$

Note that only input link faults are taken into account. Faults at the output links are considered as input link faults at the next stage.

The state probability $P\{X^{(i)} = 1\}$ of the output link is given by,

$$\begin{aligned}
P\{X^{(i)} = 1\} = \frac{1}{2}\, p_l \Big[& P\{(X^{(i+1)}, Y^{(i+1)}) = (0,1)\} \\
& + P\{(X^{(i+1)}, Y^{(i+1)}) = (1,0)\}\Big] \\
& + p_l(1 - \frac{1}{4}\, p_l) \times P\{(X^{(i+1)}, Y^{(i+1)}) = (1,1)\}
\end{aligned} \tag{3.2}$$

For a non-redundant network (as in Fig. 1) the inputs into each switch are independent. Hence,

$$P\{(X^{(i+1)}, Y^{(i+1)}) = (u,v)\} = P\{X^{(i+1)} = u\} \times P\{Y^{(i+1)} = v\} \qquad u,v = 0,1 \tag{3.3}$$

Using (3.3) and the following equation

$$P\{Y^{(i+1)} = 0\} = P\{X^{(i+1)} = 0\} = 1 - P\{X^{(i+1)} = 1\}$$

we obtain from (3.2) after some algebraic manipulations,

$$P\{X^{(i)} = 1\} = p_l \times P\{X^{(i+1)} = 1\} - \frac{1}{4}\, p_l^2 \times (P\{X^{(i+1)} = 1\})^2 \tag{3.4}$$

This expression reduces to the one derived in [4] if fault-free (i.e., $p_l = 1$) 2×2 switches are assumed.

This simple recursion formula enables us to calculate the successive state probabilities, starting from the processors outputs up to the memory inputs.
For the processors output links we have

$$P\{X^{(k)} = 1\} = p_a\, p_r \tag{3.5}$$

where p_a is the probability that a processor generates a memory request. Recursively, we calculate $P\{X^{(0)} = 1\}$. To compute the bandwidth note that the memory and its input link can be faulty as well, hence,

$$BW = N \times P\{X^{(0)} = 1\} \times p_m\, p_l \tag{3.6}$$

4. Model II

The model described in the previous section provided a lower estimate for the bandwidth of the system. The model presented in this section is more realistic, still it provides only an upper estimate for the bandwidth due to some approximations whose aim is to facilitate the analysis. This second model generalizes the analysis in [5] to the case where faulty components can be present in the system.

As in the previous section, a steady-state analysis will be performed, relying on the assumption of a large mean time between components failures. Again, the system is observed at some arbitrary yet fixed time instant t, which will be omitted from all notations for simplicity.

Since the operation of the system is synchronous, a discrete-time, discrete-space stochastic process can be used to describe the state of the system. In our case, an exact state description 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 multi-stage interconnection matrix.

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", whose aim is to make the analysis tractable, causes the calculated bandwidth to be slightly higher than its actual value, thus yielding an upper estimate for the system's bandwidth. 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 ($0 \leq R \leq N$) and thus, $N - R$ is the number of faulty processors. For a given value of R, $Z(n)$ can assume the values $0, 1, ..., R$. We denote by $P^{(R)}$ the one-step transition probability matrix whose (i,j)-th element is

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

i.e., 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 requesting memory access while $R - i$ are computing. Out of the i requests, only d ($d \leq i$) reach their destination and complete their communication. 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. Hence, $j = i + g - d$.

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 generate their memory requests independently, with a probability of p_a each. Hence,

$$G_{r,g} = \binom{r}{g} p_a{}^g (1-p_a)^{r-g} \qquad r,g = 0,...,N \quad (4.1)$$

To find $D_{i,d}$, the probability that d out of i requests will reach their final destinations, we repeat the calculation for a single stage of switches k times. 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

$$p\{a,b|u\} = \frac{\binom{N}{a} \cdot \binom{\frac{N}{2}-a}{b} \cdot 2^a}{\binom{N}{u}} \quad (4.2)$$

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 request, denoted by α_1, is hence, $\alpha_1 = p_l$. The probability of no output, 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,

$$P\{w|a\} = \binom{a}{w} \alpha_1{}^a (1-\alpha_1)^{(a-w)} \quad (4.3)$$

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 fault-free). Denote by β_2, β_1, 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,

$$P\{y,z|b\} = \binom{b}{y}\binom{b-y}{z}\beta_2{}^y\beta_1{}^z\beta_0{}^{b-y-z} \qquad (4.4)$$

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 (4.2), (4.3) and (4.4) we obtain,

$$\Phi_{u,v} = \sum_{\substack{a+2b=u \\ w+2y+z=v}} P\{a,b|u\} \cdot P\{w|a\} \cdot P\{y,z|b\} \qquad (4.5)$$

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. However, since the memories themselves may be faulty, an additional probability matrix is required. Denote by B the $(N+1)\times(N+1)$ matrix whose (v,d) element is the probability that out of v memories, exactly d will be fault-free, then,

$$B_{v,d} = \binom{v}{d} p_m{}^d (1-p_m)^{v-d} \qquad (4.6)$$

And finally, the matrix D can be obtained by

$$D = \Phi^k \cdot B \qquad (4.7)$$

The matrices G (defined in 4.1) 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,

$$P^{(R)}_{i,j} = \sum_{j=i+g-d} D_{i,d} \cdot G_{R-i+d,g} \qquad i,j = 0,...,R \qquad (4.8)$$

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

$$\Pi^{(R)} \cdot P^{(R)} = \Pi^{(R)} \quad ; \quad \sum_{i=0}^{R} \Pi_i^{(R)} = 1 \qquad (4.9)$$

On the basis of the above steady-state probabilities the bandwidth BW and processing power C 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: In a synchronous interconnection network where each blocked memory request is resubmitted,

$$BW = C \cdot \frac{p_a}{(1-p_a)} \qquad (\text{for } p_a < 1) \qquad (4.10)$$

Proof: At the beginning of a network cycle, an average number of C processors are active, joined by an average number of BW processors which have completed their memory access in the previous cycle and are now active too. Each of these $C+BW$ processors has a probability p_a of issuing a request, hence an average of $(C+BW)\times p_a$ 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. Hence, $BW = (C+BW)\times p_a$ and (4.10) follows. ∎

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

$$C(R) = \sum_{i=0}^{R} (R-i) \cdot \Pi_i^{(R)} \qquad (4.11)$$

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

$$C = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot C(R) \qquad (4.12)$$

BW can be obtained using (4.10). An alternative way of calculating BW, which must be used if $p_a = 1$, (since (4.10) does not hold in this case), is the following. Define $BW(R)$ as the conditional bandwidth for a given value of R. Then,

$$BW(R) = \sum_{i=0}^{R} \Pi_i^{(R)} \sum_{d=0}^{i} d \cdot D_{i,d} \qquad (4.13)$$

And,

$$BW = \sum_{R=0}^{N} \binom{N}{R} (p_r p_l)^R (1-p_r p_l)^{N-R} \cdot BW(R) \qquad (4.14)$$

5. Numerical Results

In this section we present some numerical comparisons between the values calculated for the bandwidth according to the two previously presented models. We expect the precise value of the bandwidth to be closer to the upper estimate than to the lower estimate which is calculated using the first (and simpler) model. Still, it is difficult to predict how different will the bandwidth be from its two estimates, without resorting to lengthy simulations. Our goal in this section is to analyze the difference between the two estimates, since a small difference will indicate that the computationally simpler model (model I) can be employed to estimate the bandwidth rather than the more complex model (Model II). Clearly, we expect the difference between the two values to be dependent on the time t, the size of the network N and the request probability p_a.

Figure 2 demonstrates the difference between the two values calculated for the bandwidth, for a system with $N = 16$ processors and 16 memories. Time is measured in $1/\lambda_r$ units. The other failure rates were chosen as $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$. The bandwidth according to the two models 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 $p_a = 0.8$) are shown in Figure 2. A very important conclusion that can be drawn from this comparison is that the difference between the two estimates is highly

dependent on the traffic as measured by p_a.

Figure 3 was plotted to study the dependence of the difference between the two calculated values on p_a, for two system sizes ($N = 16$ and $N = 32$) and two time instances ($t = 0$ and $t = 0.3$). For these four cases, the bandwidth ratio (defined as the model II value divided by the model I value) was plotted as a function of p_a. As is evident from Figure 3, the two are very close for high values of p_a. The reason for this being that when processors issue new requests at a high rate, the loss of blocked requests is negligible. Consequently, for high values of p_a, the computationally simpler model (Model I) can be used to estimate the bandwidth. Other conclusions that can be derived from Figure 3 are that the relative difference between the lower and upper estimates is high for low values of p_a, and that the value of p_a for which the bandwidth ratio is maximal is highly dependent on t, but is insensitive to the size of the network N.

To test this last observation more accurately, we have plotted in Figure 4 the bandwidth ratio as a function of N, for two values of p_a ($p_a = 0.2$, $p_a = 0.6$), and for two time instances ($t = 0$, $t = 0.3$). For a low value of p_a ($p_a = 0.2$) the bandwidth ratio is highly dependent on t and changes very slowly with N, while for a higher value of p_a ($p_a = 0.6$) the bandwidth ratio is almost independent of N.

One of the advantages of the second model is that it enables us to calculate the processing power of the multi-processor system. Figure 5 depicts the processing power of an $N \times N$ system as a function of time, for $N = 16$ and $N = 64$. To meaningfully compare systems of different sizes, we have plotted the ratio $C(t)/N$, i.e., the percentage of fault-free processors which are not idle. A very interesting (but still expected) conclusion that can be drawn from this figure is that larger systems have a lower percentage of active processors. This was observed for other values of N as well. Figure 5 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.

6. Conclusions

Two models for calculating the bandwidth of multi-processor systems with a multi-stage interconnection network in the presence of faulty components have been presented in this paper. Being able to estimate the bandwidth and the expected reduction in its value due to the occurrence of faults will allow designers to decide whether to support a graceful degradation of the system.

The first model is too pessimistic and provides an upper estimate for the bandwidth. The second model is computationally more complex than the first one but still provides only a lower estimate for the bandwidth.

The difference between these two estimates has been numerically analyzed in this paper and its dependence on the request probability p_a, the size of the network N and the time t has been studied.

7. 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," to appear, *Algorithm, Architecture and Technology Issues in Models of Concurrent Computations,* S. Tewksbury (ed.), Plenum, 1988.

[3] 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.

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

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

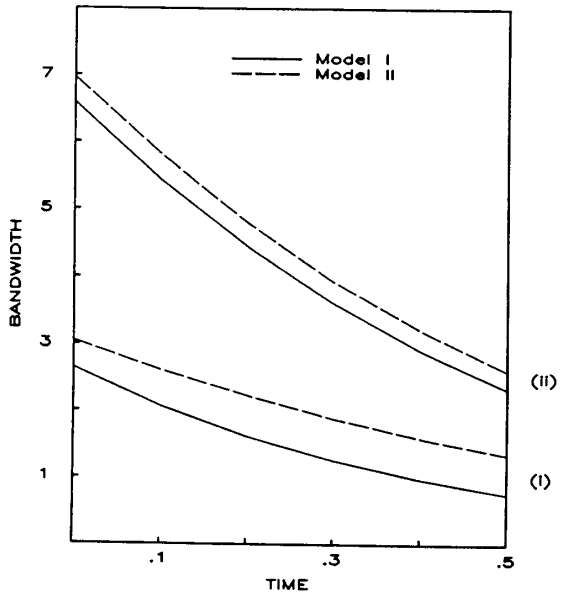

Figure 2: The bandwidth of a 16×16 network as a function of time for Model I and Model II and (*i*) $p_a = 0.2$ and (*ii*) $p_a = 0.8$. (Failure rates: $\lambda_m/\lambda_r = 0.7$ and $\lambda_l/\lambda_r = 0.2$.)

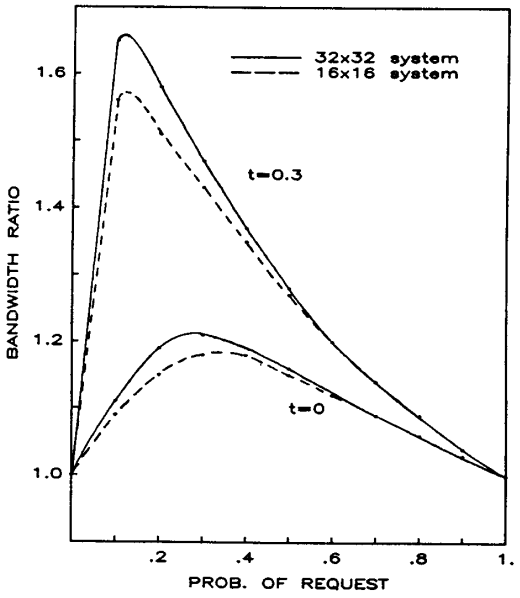

Figure 3: Bandwidth ratio (Model II *BW* / Model I *BW*) for two multi-stage networks as a function of p_a for two different values of t.

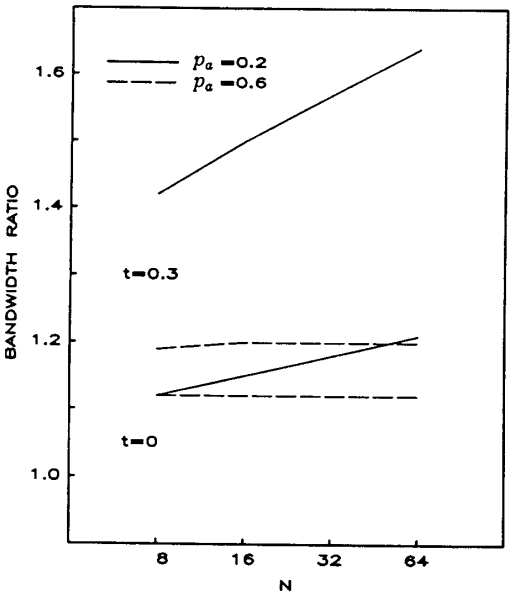

Figure 4: Bandwidth ratio (Model II *BW* / Model I *BW*) for two values of p_a and two values of t as a function of N.

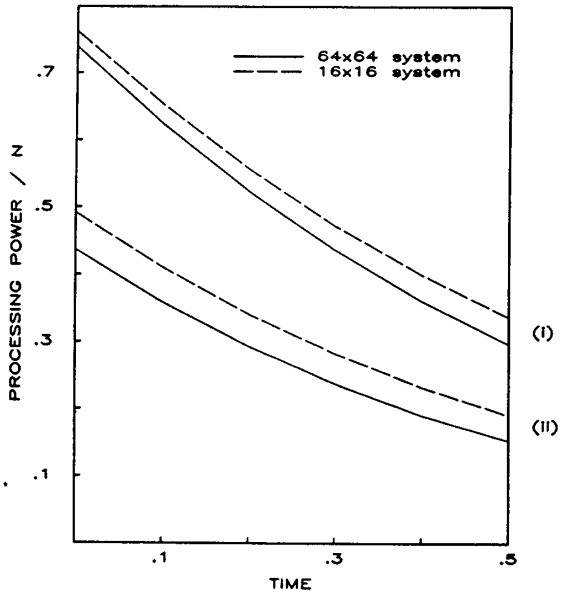

Figure 5: The processing power of two $N \times N$ networks as a function of time for (*i*) $p_a = 0.2$ and (*ii*) $p_a = 0.4$.