Probabilistic Analysis of a Crossbar Switch[1]

by

T. N. Mudge and B. A. Makrucki
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109

Abstract

This paper presents a probabilistic analysis of a crossbar switch interconnection network. A crossbar switch can be used to interconnect various combinations of computer subsystems. In the analysis below it is assumed, without loss of generality, that the crossbar is being used to connect N processors to M memories. The crossbar is termed an N-M crossbar (read "N to M crossbar"). General expressions are developed for a variety of performance figures for an N-M crossbar including: the probability of a memory request being accepted (i.e. not being blocked by another request to the same memory), the expected bandwidth of the crossbar, and the average wait time of a request before it is accepted. Closed form solutions to these expressions are given for the uniform request case and for the favorite memory case (i.e. where processor i requests memory i with a higher probability than others memories). The closed form solutions are tested against simulations.

1. Introduction

This paper presents a probabilistic analysis of a crossbar switch interconnection network. It is based on work reported in [MaM81b]. A crossbar switch can be used to interconnect various combinations of computer subsystems including processors to processors, processors to memories, processors to I/O devices, and memories to I/O devices. In the analysis below it is assumed, without loss of generality, that the crossbar is being used to connect N processors to M memories. This multiprocessor system is depicted in Figure 1. The crossbar is termed an N-M crossbar (read "N to M crossbar"). In the operation of the system it is further assumed that their is a system wide clock, and that read and write memory requests made by the processors can only occur in synchronism with this clock.

The designer of a multiprocessor system has a wide variety of interconnection networks to choose from. An obvious candidate for an interconnection network is a shared bus--several processors and memories connected to a shared time-multiplexed bus. However, a shared bus only provides high speed interconnection if it is very fast relative to processors and memories. If it is desired to connect N processors to M memories a shared bus' performance decays as N increases. To improve performance over the shared bus a crossbar switch may be used to establish multiple bus connections between processors and memories. A crossbar allows any set of simultaneous interconnections in which at most one input bus is connected to each output bus. Unlike the shared bus its performance does not decrease with increase in N. However, the component complexity of the crossbar grows as $O(NM)$. For this reason past proposals for tightly coupling multiple processors to multiple memories have steered away from crossbar switches. In their place a whole range of ingenious networks have been proposed that include the following: the networks of Clos and Benes [Clo53,Ben65], the Banyan networks of Goke and Lipovski [GoL73], the Data Manipulator network of Feng [Fen74], the Omega network of Lawrie [Law75], the STARAN flip network of Batcher [Bat76], the Indirect Binary n-cube of Pease [Pea77], the Generalized Cube of Siegel and Smith [SiS78, SiM81b], the Delta network of Patel [Pat79], the Baseline network of Wu and Feng [WuF80], and the Augmented Data Manipulator network of Siegel and McMillen [SiM81a]. These networks have many of the connectivity properties of a crossbar without the component complexity. For a good survey of the state-of-the-art in networks see Siegel [Sie80]. A unified theory developed by Lipovski and Malek that characterizes the inter-relationship of many of these networks will appear in [LiM]. With the exception of some forms of the of the Banyan network the component complexity of these networks all grow as $O(Nlog_2N)$ rather than $O(N^2)$ (assuming $N=M$).

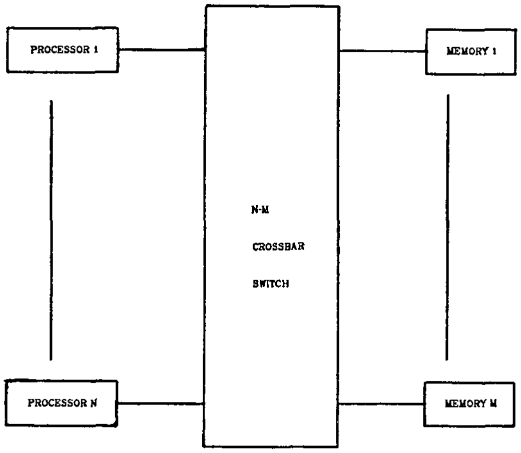


Figure 1. Multiprocessor System.

In the context of VLSI technology it is no longer clear that reduced component complexity is an advantage **within** a single IC. For example, preliminary layouts for a Delta network and a bit-slice crossbar carried out by us [MaM81a] suggest that the reduced complexity networks do not appear to translate into more efficient space utilization in an IC layout. The decreased importance of component complexity or device count as

[1] This research was supported by the National Science Foundation under Grant MCS-8009315.

a measure of design efficiency in ICs has been discussed by Thompson and Franklin [Tho80, Fra80] among others. Furthermore, although the reduced complexity networks preserve some of the connectivity properties of a crossbar they do not preserve bandwidth [Pat79]. For these reasons we have decided to explore more fully the design and analysis of crossbar switches for use as single chip building blocks in interconnection networks. However, it is still assumed that if the size requirements of an interconnection network exceed the size of the single chip crossbar, the network will be constructed as an "$N\log_2 N$" type network since the reduced component complexity argument holds at the IC package level.

Previous work that is relevant to this paper can be found in [Rav72, Str70, Bha75, Pip75, BaS76, CKL77, Hoo77, Rau79, Pat81]. The analysis presented below is an extension of this work. Analyses presented in the above references assume that processor requests are uniformly distributed among the memories. Our analysis relaxes this condition. In particular, we allow the possibility of a processor not making a memory request during a system cycle[2], and we allow the requests to be non-uniformly distributed among the memories. In the case of each processor having a favorite memory, i.e., being more likely to request one particular memory than any other, a new closed form solution for the probability of request acceptance is developed. In contrast to some of the analyses presented in [BaS76, Bha75, CKL77, Rau79] we have not attempted to model dependencies between successive memory requests or to model the effect of being allowed to queue memory requests, although an expression for the expected wait time of a memory request is developed in section 4. Including dependencies or queues in the analysis results in very complex models that, although more accurate, can only be solved for cases where all processors make uniform requests every system cycle (see [Rau79]).

2. Analysis of an N-M Crossbar

Assumptions: Consistent with the system behavior outlined in the Introduction it is assumed that events in the system occur at discrete time intervals defined by the system clock. The discrete time intervals are termed system cycles. Memory requests made by the processors occur synchronously with the system clock and any processor may make a memory request at the start of any system cycle subject to the probability distributions defined below. A memory can service one request per system cycle.

Definition 1: Let R_i be the event that processor i requests any memory.

Definition 2: Let S_{ij} be the event that processor i requests memory j.

Assumptions: Assume the request behavior of a processor is an independent process. Assume also that the request behavior of each processor is independent of that of any other processor. We shall term these assumptions the independent request property (IRP).

One consequence of the IRP is that our analysis does not model resubmission of denied requests. In other words if several processors request the same memory those that do not get the memory (all but one) "loose" their request, and in the next time period their requests will be made without regard to this loss since the request behavior of a processor is an independent process. In a realistic situation the request would most likely be resubmitted. The effect of this shortcoming of the analysis is measured experimentally, and the results are prescribed in a later section.

One other consequence of the IRP is that the following holds:

$$Pr\{S_{ab} \cap S_{cd}\} = Pr\{S_{ab}\}Pr\{S_{cd}\} \quad \text{for all } a \neq c$$

Assumptions: Assume that it is possible to measure the following two statistics empirically.

(1) The probability, r_i, that processor i requests any memory at the start of a system cycle.

(2) The probability, $p_i(j)$, that processor i requests memory j given that it makes any request at all at the start of a system cycle.

The values of r_i and the $p_i(j)$ for a particular processor i will be termed the request distributions for that processor.

The statistic r_i and the $p_i(j)$ may be estimated from memory reference counts obtained from typical programs (see [BaS76, Hoo77]).

From definitions 1 and 2 and the above assumptions it follows that:

$$r_i = Pr\{R_i\} \tag{1}$$

$$p_i(j) = Pr\{S_{ij} \mid R_i\} \tag{2}$$

From the definition of conditional probability we have:

$$Pr\{S_{ij} \mid R_i\} = \frac{Pr\{S_{ij} \cap R_i\}}{Pr\{R_i\}} \qquad (Pr\{R_i\}>0)$$

However, from definitions 1 and 2 it follows that:

$$S_{ij} \cap R_i = S_{ij}$$

(See event space diagrams in Figure 2.)

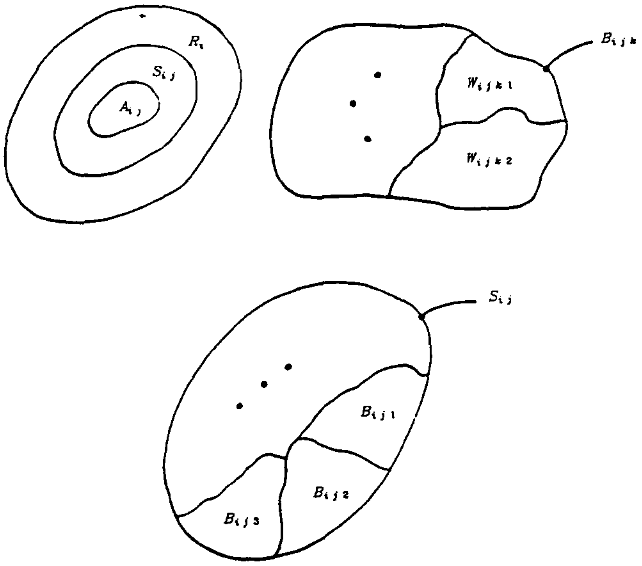


Figure 2. Event Space Diagrams.

[2] See next section for a definition of this term.

Therefore:

$$Pr\{S_{ij}\} = Pr\{S_{ij} \,|\, R_i\}\; Pr\{R_i\}$$

Using (1) and (2) this can be written:

$$Pr\{S_{ij}\} = r_i p_i(j) \qquad (3)$$

Definition 3: Let A_{ij} be the event that processor i requests memory j **and** is accepted.

Definition 4: Let B_{ijk} be the event that processor i and $k-1$ others request memory j.

Assumption: If exactly k processors request memory j assume that the one processor whose request is serviced is selected from the set of k with a probability of $\frac{1}{k}$. We shall term this the unform selection rule (USR). In the notation developed so far the USR can be expressed as follows:

$$Pr\{A_{ij} \,|\, B_{ijk}\} = \frac{1}{k}$$

From definitions 2 and 3 it follows that:

$$A_{ij} \cap S_{ij} = A_{ij}$$

(See event space diagrams in Figure 2.)

Furthermore, from definitions 2 and 4 it follows that:

$$S_{ij} = \bigcup_{k=1}^{N} B_{ijk}$$

(See event space diagrams in Figure 2.)

Combining these two observations gives the following:

$$A_{ij} = A_{ij} \cap S_{ij} = A_{ij} \cap \left[\bigcup_{k=1}^{N} B_{ijk} \right]$$

Distributing gives:

$$A_{ij} = \bigcup_{k=1}^{N} \left[A_{ij} \cap B_{ijk} \right] \qquad (4)$$

From definition 4 it follows that the events B_{ijk} are mutually exclusive with respect to k, i.e. $B_{ijk_1} \cap B_{ijk_2} = \emptyset$ for all $k_1 \neq k_2$. Therefore, the events $\left[A_{ij} \cap B_{ijk} \right]$ are mutually exclusive with respect to k also. This fact allows us to express the relationship between the probabilities associated with the events in equation 4 as follows:

$$Pr\{A_{ij}\} = \sum_{k=1}^{N} Pr\{A_{ij} \cap B_{ijk}\} \qquad (5)$$

Using the definition of conditional probability again, we have:

$$Pr\{A_{ij} \cap B_{ijk}\} = Pr\{A_{ij} \,|\, B_{ijk}\}\; Pr\{B_{ijk}\}$$

Substituting this in 5 gives:

$$Pr\{A_{ij}\} = \sum_{k=1}^{N} Pr\{A_{ij} \,|\, B_{ijk}\}\; Pr\{B_{ijk}\}$$

Applying the USR assumption to this gives:

$$Pr\{A_{ij}\} = \sum_{k=1}^{N} \frac{1}{k}\; Pr\{B_{ijk}\} \qquad (6)$$

Definition 5: Define a binary vector, $\mathbf{w}_{ijk} = \langle w_{ijk}(1),\ldots,w_{ijk}(N) \rangle$ to represent the request pattern seen by memory j when processors i and $k-1$ others request memory j. The elements of $\mathbf{w}_{ijk}$ are defined as follows:

$$w_{ijk}(h) = 1$$

iff processors h, i, and $k-2$ others request memory j.

$$w_{ijk}(h) = 0$$

iff processor h makes no request or requests a memory other than j, and processors i and $k-1$ others request memory j.

There are $\binom{N-1}{k-1}$ distinct vectors $\mathbf{w}_{ijk}$. When necessary they will be distinguished by an additional subscript as follows:

$$\mathbf{w}_{ijkl} \qquad l = 1,\ldots,\binom{N-1}{k-1}$$

From definition 5 it follows that:

$$w_{ijkl}(i) = 1$$

$$\sum_{h=1}^{N} w_{ijkl}(h) = k$$

and

$$\{w_{ijkl}(h) = 1\} \Rightarrow \textit{event } S_{hj} \qquad (7)$$

However, the converse of relation 7 does not necessarily hold since the event that processor h requests memory j says nothing about processor i requesting memory j nor does it indicate how many processors request memory j.

Definition 6: Denote the occurrence of event S and its non-occurrence by S^1 and S^0 respectively.

From definitions 5 and 6 and relation 7 we have:

$$\mathbf{w}_{ijkl} \Rightarrow \bigcap_{h=1}^{N} S_{hj}^{w_{ijkl}(h)}$$

Furthermore, the converse holds since the right hand side completely specifies the request pattern seen by memory j. Therefore, we can write:

$$\mathbf{w}_{ijkl} \Leftrightarrow \bigcap_{h=1}^{N} S_{hj}^{w_{ijkl}(h)} \qquad (8)$$

Definition 7: Let W_{ijkl} be the event corresponding to the request pattern $\mathbf{w}_{ijkl}$.

From relation 8 and definition 7 it follows that:

$$W_{ijkl} = \bigcap_{h=1}^{N} S_{hj}^{w_{ijkl}(h)}$$

Applying the IRP assumptions to this allows us to express the relationship between the probabilities associated with the events as follows:

$$Pr\{W_{ijkl}\} = \prod_{h=1}^{N} Pr\{S_{hj}^{w_{ijkl}(h)}\} \qquad (9)$$

From equation 3 and definition 6 it follows that:

$$Pr\{S_{hj}^{1}\} = r_h p_h(j)$$

and

$$Pr\{S_{hj}^{0}\} = 1 - r_h p_h(j)$$

Substituting these results into equation 9 gives:

$$Pr\{W_{ijkl}\} = \prod_{h=1}^{N} [r_h p_h(j)]^{w_{ijkl}(h)} [1 - r_h p_h(j)]^{1 - w_{ijkl}(h)}$$

However, as noted earlier from definition 5 it follows that:

$$w_{ijkl}(i) = 1$$

Thus the above equation can be rewritten to give:

$$Pr\{W_{ijkl}\} = r_i p_i(j) \prod_{\substack{h=1 \\ h \neq i}}^{N} [r_h p_h(j)]^{w_{ijkl}(h)} [1 - r_h p_h(j)]^{1-w_{ijkl}(h)} \qquad (10)$$

From definition 4, 5 and 7 it follows that:

$$B_{ijk} = \bigcup_{l=1}^{\binom{N-1}{k-1}} W_{ijkl} \qquad (11)$$

(See event space diagrams in Figure 2.)

Furthermore, from definitions 5 and 7 it may be concluded that events W_{ijkl} are mutually exclusive with respect to l, i.e. $W_{ijkl_1} \cap W_{ijkl_2} = \emptyset$ for all $l_1 \neq l_2$. This allows us to express the relationship between the probabilities associated with the events in equation 11 as follows:

$$Pr\{B_{ijk}\} = \sum_{l=1}^{\binom{N-1}{k-1}} Pr\{W_{ijkl}\} \qquad (12)$$

Substituting from equation 10 into 12 gives:

$$Pr\{B_{ijk}\} = r_i p_i(j) \sum_{l=1}^{\binom{N-1}{k-1}} \prod_{\substack{h=1 \\ h \neq i}}^{N} [r_h p_h(j)]^{w_{ijkl}(h)} [1 - r_h p_h(j)]^{1-w_{ijkl}(h)} \qquad (13)$$

Substituting equation 13 into equation 6 gives us the following expression for the probability of processor i successfully requesting memory j:

$$Pr\{A_{ij}\} = r_i p_i(j) \sum_{k=1}^{N} \frac{1}{k} \sum_{l=1}^{\binom{N-1}{k-1}} \prod_{\substack{h=1 \\ h \neq i}}^{N} [r_h p_h(j)]^{w_{ijkl}(h)} [1 - r_h p_h(j)]^{1-w_{ijkl}(h)} \qquad (14)$$

From the point of view of individual processors an interesting quantity is the probability of processor i successfully requesting memory j once i has requested j, i.e. $Pr\{A_{ij} | S_{ij}\}$. This gives a measure of a request not being blocked by a conflict. To obtain $Pr\{A_{ij} | S_{ij}\}$ we need to modify equation 14 slightly. This can be done in a straightforward manner as follows. Above it was observed that definitions 2 and 3 lead to:

$$A_{ij} \cap S_{ij} = A_{ij}$$

Combining this with the definition of conditional probability gives:

$$Pr\{A_{ij} | S_{ij}\} = \frac{Pr\{A_{ij} \cap S_{ij}\}}{Pr\{S_{ij}\}} = \frac{Pr\{A_{ij}\}}{Pr\{S_{ij}\}}$$

Substituting for $Pr\{S_{ij}\}$ from equation 3 and for $Pr\{A_{ij}\}$ from equation 14 gives :

$$Pr\{A_{ij} | S_{ij}\} = \sum_{k=1}^{N} \frac{1}{k} \sum_{l=1}^{\binom{N-1}{k-1}} \prod_{\substack{h=1 \\ h \neq i}}^{N} [r_h p_h(j)]^{w_{ijkl}(h)} [1 - r_h p_h(j)]^{w_{ijkl}(h)} \qquad (15)$$

In general, it is desirable that an interconnection network provide as many simultaneous channels as needed from its N inputs to its M outputs. In the case of an N-M crossbar the number of simultaneous channels needed is given by:

$$Req[BW] \overset{\Delta}{=} \sum_{i=1}^{N} r_i \qquad (16)$$

The left hand side reads "requested bandwidth". It is measured in "channels", however, if the data rate per channel can be expressed in Hertz then it is possible to represent $Req[BW]$ in Hertz; units more commonly used to measure bandwidth. Two other bandwidth measures are useful for characterizing an N-M crossbar. They are defined as follows:

$$E[BW] \overset{\Delta}{=} \sum_{i=1}^{N} \sum_{j=1}^{M} Pr\{A_{ij}\} \qquad (17)$$

$$Max[BW] \overset{\Delta}{=} min(N, M) \qquad (18)$$

The first of these is just the expected bandwidth, i.e. the expected number of channels in use between the N processors and the M memories. The second definition is a measure of the maximum number of channels that can exist between the processors and memories if request conflicts never occur. It can be seen that $Max[BW]$ depends only on the structure of the N-M crossbar, whereas $Req[BW]$ and $E[BW]$ are also dependent on the request distributions that define each processor's request behavior (i.e. r_i and $p_i(j)$).

Using the definitions given in equations 16, 17, and 18 we can define two figures of merit for an N-M crossbar operating under a particular set of request distributions as follows:

$$E \overset{\Delta}{=} \frac{E[BW]}{Req[BW]} \qquad (19)$$

$$U \overset{\Delta}{=} \frac{E[BW]}{Max[BW]} \qquad (20)$$

Where E is a measure of the *effectiveness* of the crossbar in fulfilling the demands of the particular set of request distributions, and U is a measure of the *utilization* of the crossbar given the particular set of request distributions. In general, E and U are both functions of N, M, r_i and $p_i(j)$ for all i and j. Also, $0 \leq E \leq 1$ and $0 \leq U \leq 1$.

3. Closed Form Solutions

For certain functional forms for r_i and $p_i(j)$ it is possible to derive closed form expressions for equation 14, and as a consequence equations 15 through 20 too. In this section we shall illustrate this possibility with several examples.

3.1. Uniform Request Distributions

In this case:

let $r_i = r$ for all i and j
and $p_i(j) = \frac{1}{M}$ for all i,

Then equation 14 simplifies to the following:

$$Pr\{A_{ij}\} = \frac{r}{M} \sum_{k=1}^{N} \frac{1}{k} \sum_{l=1}^{\binom{N-1}{k-1}} \prod_{\substack{h=1 \\ h \neq i}}^{N} [\frac{r}{M}]^{w_{ijkl}(h)} [1 - \frac{r}{M}]^{1-w_{ijkl}(h)} \qquad (21)$$

Recall that for all values of l exactly k elements of $\mathbf{w}_{ijkl}$ are 1 ($k-1$ elements if the $h=i$ case is omitted), and $N-k$ elements are 0. Therefore, equation 21 reduces to:

$$Pr\{A_{ij}\} = \frac{r}{M} \sum_{k=1}^{N} \frac{1}{k} \sum_{l=1}^{\binom{N-1}{k-1}} [\frac{r}{M}]^{k-1}[1-\frac{r}{M}]^{N-k} \qquad (22)$$

Since l is no longer a parameter of any of the terms in the scope of the rightmost summation, equation 22 reduces to:

$$Pr\{A_{ij}\} = \frac{r}{M} \sum_{k=1}^{N} \frac{1}{k} \binom{N-1}{k-1} [\frac{r}{M}]^{k-1}[1-\frac{r}{M}]^{N-k}$$

This can be written as:

$$Pr\{A_{ij}\} = \frac{1}{N} \sum_{k=1}^{N} \binom{N}{k} [\frac{r}{M}]^{k}[1-\frac{r}{M}]^{N-k} \qquad (23)$$

Recalling the binomial expansion of $[\frac{r}{M} + (1-\frac{r}{M})]^N$ we can rewrite 23 as follows:

$$Pr\{A_{ij}\} = \frac{1}{N} [1-(1-\frac{r}{M})^N] \qquad (24)$$

Also:

$$Pr\{A_{ij} \mid S_{ij}\} = \frac{M}{rN} [1-(1-\frac{r}{M})^N] \qquad (25)$$

Equations 16 through 18 simplify to:

$$\begin{aligned} Req[BW] &= rN \\ E[BW] &= M[1-(1-\frac{r}{M})^N] \qquad (26) \\ Max[BW] &= min(N,M) \end{aligned}$$

Therefore equations 19 and 20 simplify to:

$$\begin{aligned} E &= \frac{M}{rN}[1-(1-\frac{r}{M})^N] \\ U &= Max\{[1-(1-\frac{r}{M})^N], \frac{M}{N}[1-(1-\frac{r}{M})^N]\} \end{aligned} \qquad (27)$$

Further simplifying assumptions lead to even more concise closed forms as follows:

Let $\alpha = \frac{N}{M}$

Then equation 25 has a limiting form as follows:

$$Pr\{A_{ij} \mid S_{ij}\} = \frac{1}{r\alpha} [1-(1-\frac{r\alpha}{N})^N]$$

If N and M grow large this last equation has a limiting form as follows:

$$\lim_{N\to\infty} \frac{1}{r\alpha} [1-(1-\frac{r\alpha}{N})^N] = \frac{1}{r\alpha} [1-e^{-r\alpha}] \qquad (28)$$

Most of the equations 24 through 28 have previously been derived in [Str70, CKL77, BrD77] for the case $r=1.0$, and in [Hoo77, Pat81] for the case $r \leq 1.0$.

Figures 3 through 5 illustrate some of the results obtained by assuming uniform request distributions. Figure 3 shows $Pr\{A_{ij} \mid S_{ij}\}$ and $E[BW]$ for a 4-M crossbar as a function of M when $r=0.1$ (i.e. a 10% duty cycle for memory requests). As one would expect the probability of a submitted request being accepted approaches 1 as M increases, since increasing M while keeping N constant reduces the likelihood of requests conflicting for memories. For the same reason the expected bandwidth in use, $E[BW]$ approaches the expected bandwidth requested, $Req[BW]$. (Recall that $Req[BW]=rN=0.4$.) Figure 4 again shows $Pr\{A_{ij} \mid S_{ij}\}$ and $E[BW]$ for a 4-M crossbar as a function of M, however, the request has been increased to the point where every processors makes a request at every memory cycle, i.e. $r=1.0$. Again $Pr\{A_{ij} \mid S_{ij}\}$ asymptotically approaches 1 with increase in M, but at a slower rate than before. In addition, $E[BW]$ approaches $Req[BW]$ (=4), however, since

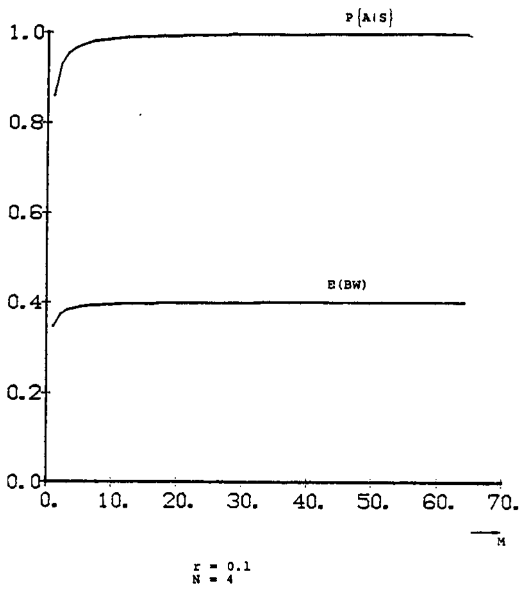


Figure 3.

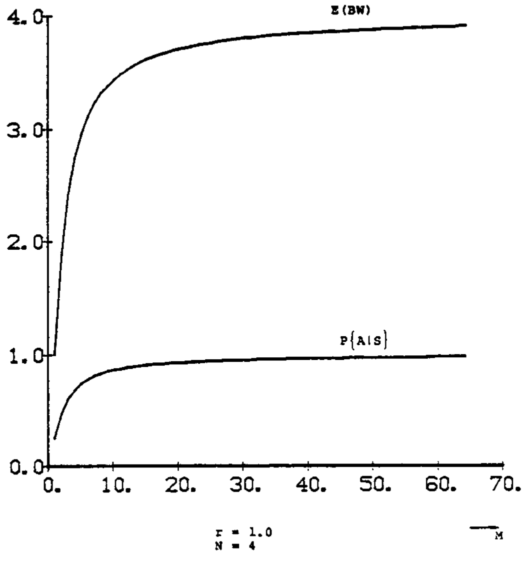


Figure 4.

$r=1.0$, $E[BW]$ also approaches the structural limit of the crossbar, namely $Max[BW]$ (=4). Figure 5 illustrate how $E[BW]$ varies with r for a 4-4 crossbar, a 16-16 crossbar, and a 64-64 crossbar. Notice that in all three cases 65-70% of the maximum number of channels available are being used when $r=1.0$.

3.2. Favorite Memory Distributions

In this case assume that processor i is more likely to request memory i than other memories. To model this case let:

$$\begin{array}{ll} r_i = r & \text{for all } i \\ p_i(i) = p & \text{for all } i \\ p_i(j) = \dfrac{1-p}{M-1} & \text{for all } i \neq j \end{array}$$

For i to be the favorite we require that $p > \frac{1}{M}$. For brevity and to simplify the analysis let $N=M$. The cases $N>M$ and $N<M$ have closed forms that can be obtained using a straightforward extension of the analysis presented below. In deriving an expression for $Pr\{A_{ij} \mid S_{ij}\}$ two cases arise:

Case 1: $i=j$

Equation 15 simplifies to the following:

$$Pr\{A_{ii} \mid S_{ii}\} = \sum_{k=1}^{N} \frac{1}{k} \sum_{l=1}^{\binom{N-1}{k-1}} [\frac{r(1-p)}{M-1}]^{k-1}[1-\frac{r(1-p)}{M-1}]^{N-k}$$

For convenience let:

$$\frac{r(1-p)}{M-1} = t$$

Then:

$$Pr\{A_{ii} \mid S_{ii}\} = \sum_{k=1}^{N} \frac{1}{k} \binom{N-1}{k-1} t^{k-1}(1-t)^{N-k}$$

Recalling the binomial expansion of $[t+(1-t)^N]$, allows us to write:

$$Pr\{A_{ii} \mid S_{ii}\} = \frac{1}{Nt}[1-(1-t)^N] \qquad (29)$$

As would be expected the expression on the right hand side of equation 29 reaches a maximum as $p \rightarrow 1$. Indeed, using l'Hopital's rule it can be shown that:

$$\lim_{p \rightarrow 1} Pr\{A_{ii} \mid S_{ii}\} = 1$$

Case 2: $i \neq j$

In this case the set of possible request patterns seen by memory j partition into two mutually exclusive subsets. These subsets are distinguished by whether or not each of their patterns include a request from processor j (a favorite request). Let $\{\mathbf{w}_{ij**}\}$ be the set of possible request patterns where * indicates that the subscript in that position ranges over its allowed values. Then the set of possible request patterns that each include a request from processor j is given by:

$$\alpha = \{ \mathbf{w}_{ij**} \mid \mathbf{w}_{ijkl}(j)=1 \}$$

And the set of possible request patterns none of which include a request from processor j is given by:

$$\beta = \{ \mathbf{w}_{ij**} \mid \mathbf{w}_{ijkl}(j)=0 \}$$

Examination of sets α and β reveals that:

$$|\alpha| = \binom{N-2}{k-2} \quad 2 \leq k \leq N \qquad (30)$$

and

$$|\beta| = \binom{N-2}{k-1} \quad 1 \leq k \leq N \qquad (31)$$

To see this, notice that for set α processors i and j request memory j leaving $k-2$ other processors out of a total of $N-2$ as possible requestors. Also, notice that for set β processor i requests memory j leaving $k-1$ other processors out of a total of $N-2$ as possible requestors. The total of $N-2$ is obtained by noting that for both sets α and β the requests from processors i and j are fixed.

An event corresponding to any $\mathbf{w}_{ijkl} \; \varepsilon \; \alpha$ occurs with the following probability:

$$rpt^{k-1}[1-t]^{N-k} \qquad (32)$$

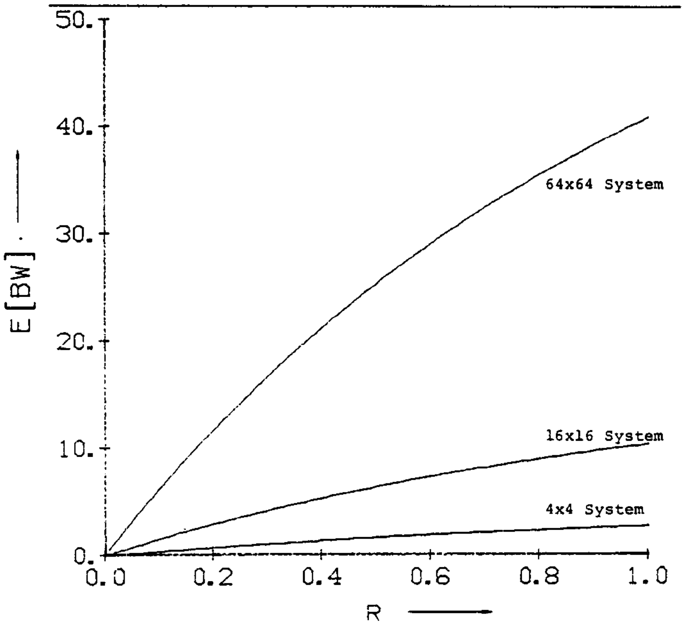
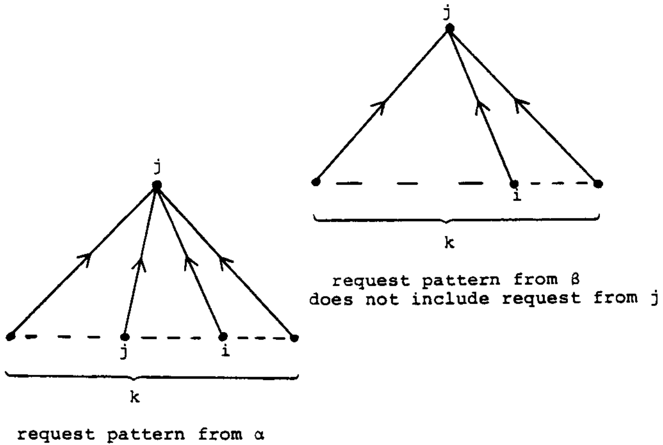


Figure 5.



request pattern from β
does not include request from j

request pattern from α
includes request from j

Figure 6. Examples of Request Patterns.

since processor j requests its favorite memory, j, with probability rp; $k-1$ other processors (including i) make non-favorite requests to memory j with probability t^{k-1}; and the $N-k$ remaining processors do not request memory j with probability $[1-t]^{N-k}$.

Similarly, an event corresponding to any $w_{ijkl} \varepsilon \beta$ occurs with probability:

$$(1-rp)t^k(1-t)^{N-k-1} \tag{33}$$

Since processor j does not request memory j with probability $(1-rp)$; k other processors (including i) make non-favorite requests to j with probability t^k; and the $N-k-1$ remaining processors do not request memory j with probability $(1-t)^{N-k-1}$.

Using the results of equations 30, 31, 32, 33, together with the fact that in the case we are considering $Pr\{S_{ij}\} = r_i p_i(j) = t$, allows us to simplify equation 15 to the following:

$$\begin{aligned} Pr\{A_{ij} \mid S_{ij}\} = &\sum_{k=2}^{N} \frac{1}{k} \binom{N-2}{k-2} rpt^{k-2}(1-t)^{N-k} \\ &+ \sum_{k=1}^{N} \frac{1}{k} \binom{N-2}{k-1}(1-rp)t^{k-1}(1-t)^{N-k-1} \end{aligned} \tag{34}$$

The summation over k splits into two parts corresponding to the mutually exclusive events associated with sets α and β.

Equation 34 can be reduced to a closed form using some simple algebraic identities. The steps involved are outlined below:

Recall
$$\binom{N-2}{k-2} = \binom{N-1}{k-1} - \binom{N-2}{k-1} \tag{35}$$

This equation holds for all integer values of k provided we adopt the convention that the binomial coefficient $\binom{N}{k}=0$ if $k<0$ or if $k>N$ (see Knuth p.53 [Knu68]). Equation 35 allows us to rewrite equation 34 as follows:

$$\begin{aligned} Pr\{A_{ij} \mid S_{ij}\} = &\sum_{k=2}^{N} \frac{1}{k} \binom{N-1}{k-1} rpt^{k-2}(1-t)^{N-k} \\ &- \sum_{k=2}^{N-1} \frac{1}{k} \binom{N-2}{k-1} rpt^{k-2}(1-t)^{N-k} \\ &+ \sum_{k=1}^{N-1} \frac{1}{k} \binom{N-2}{k-1}(1-rp)t^{k-1}(1-t)^{N-k-1} \end{aligned}$$

Further manipulation of the binomial coefficients gives:

$$\begin{aligned} Pr\{A_{ij} \mid S_{ij}\} = &\frac{rp}{Nt^2} \sum_{k=2}^{N} \binom{N}{k} t^k(1-t)^{N-k} \\ &- \frac{rp(1-t)}{(N-1)t^2} \sum_{k=2}^{N-1} \binom{N-1}{k} t^k(1-t)^{N-k-1} \\ &+ \frac{(1-rp)}{(N-1)t} \sum_{k=1}^{N-1} \binom{N-1}{k} t^k(1-t)^{N-k-1} \end{aligned}$$

Recalling the binomial expansion of $[t+(1-t)]^N$ enables us to arrive at the following closed form:

$$\begin{aligned} Pr\{A_{ij} \mid S_{ij}\} = &\frac{rp}{Nt^2} [1-(1-t)^N - Nt(1-t)^{N-1}] \\ &- \frac{rp(1-t)}{(N-1)t^2} [1-(1-t)^{N-1} - (N-1)t(1-t)^{N-2}] \\ &+ \frac{(1-rp)}{(N-1)t} [1-(1-t)^{N-1}] \end{aligned}$$

canceling terms reduces this to:

$$\begin{aligned} Pr\{A_{ij} \mid S_{ij}\} = &\frac{rp}{Nt^2} [1-(1-t)^N] \\ &+ \frac{t-rp}{(N-1)t^2} [1-(1-t)^{N-1}] \end{aligned} \tag{36}$$

Together equations 29 and 36 define $Pr\{A_{ij} \mid S_{ij}\}$ for all values of i and j in the case where processor i requests memory i as a favorite memory. Recall that for memory i to be a favorite $p>\frac{1}{M}$, and that $\lim_{p \to 1} Pr\{A_{ii} \mid S_{ii}\} = 1$. In addition, it can be shown, using l'Hopital's rule with equation 36, that $\lim_{p \to 1} Pr\{A_{ij} \mid S_{ij}\} = 0$ for $i \neq j$. This suggests that in the case of favorite memories the USR be abandoned and the requests from non-favorite processors be given higher priority than those from the favorite processor.

Furthermore, it can be shown that: $E[BW]_{favorite} > E[BW]_{uniform}$. Finally, in the limiting case of $p=1$ it can be shown that: $E[BW]_{favorite} = min(N,M)$. This agrees with our definition of $Max[BW]$ (see equation 15), which is consistent because in the case of $p=1$ all requests are exclusively between processors and their favorite memories, i.e. no conflicts occur.

4. Simulation Results

Simulations were run in an effort to compare the analytical results with a more realistic system operation. It was found that the IRP assumptions, although not realistic, did not introduce significant differences.

The simulator is designed to model a system where processors emit requests according to their respective request distributions. Furthermore, once requests are emitted they must wait for a connection. That is, if processor i emits a request for memory j and it is not accepted the request is resubmitted on every succeeding cycle until the connection is granted. Each memory has associated with it N request fields where processors store requests. During each cycle each memory selects the requesting processor that gets the connection uniformly from all requests in its set of N request fields (empty request fields are ignored). The simulation results are shown in the Appendix.

Definition 8: Let $N_W(i,j)$ be the number of cycles that processor i spends waiting for a requested connection to memory j.

Definition 9: Let $N_R(i,j)$ be the number of cycles that processor i requests or uses memory j (i.e. it is the number of cycles during which processor i references memory j).

Definition 10: Let $WF(i,j)$ be the waiting fraction for processor i requesting memory j. That is, it is the fraction of cycles that processor i spends waiting for memory j, relative to the number of requests processor i generates for memory j.

Then, from definitions 8, 9, and 10 it follows that:

$$WF(i,j) = \frac{N_W(i,j)}{N_R(i,j)}$$

One would expect that:

$$WF(i,j) \approx 1 - Pr\{A_{ij} \mid S_{ij}\}$$

That is, the number of cycles spent waiting is approximately the probability of not having a request accepted, multiplied by the number of references to that memory.

From simulation runs for many request distributions this approximation has been found to hold within about 10%.

Now notice that if for example $WF(i,j) = \frac{2}{3}$, then it may be seen that the average waiting time is 2 while the number of requests is 3. That is, for each request for memory j, on the average processor i must wait for 2 cycles and then use the connection on the third.

Definition 11: Let $E[W(i,j)]$ be the average waiting time of processor i for memory j. Then, from the above discussion it follows that:

$$\frac{E[W(i,j)]}{E[W(i,j)]+1} = WF(i,j) \approx 1 - Pr\{A_{ij} \mid S_{ij}\}$$

So,

$$E[W(i,j)] \approx \frac{1 - Pr\{A_{ij} \mid S_{ij}\}}{Pr\{A_{ij} \mid S_{ij}\}}$$

From this simple approximate analysis the average waiting time may be found. Note some special cases:

$$Pr\{A_{ij} \mid S_{ij}\} = \frac{1}{2} \quad \Rightarrow E[W(i,j)] = 1$$

$$Pr\{A_{ij} \mid S_{ij}\} = \frac{1}{3} \quad \Rightarrow E[W(i,j)] = 2.$$

The limiting behavior of the waiting time is:

$$\lim_{N\to\infty} E[W] = \frac{1-\frac{1}{r\alpha}\;[1-e^{-r\alpha}]}{\frac{1}{r\alpha}\;[1-e^{-r\alpha}]} = \frac{r\alpha}{1-e^{-r\alpha}} - 1$$

If $r\alpha = 1$, then $\lim_{N\to\infty} E[W] = \frac{1}{e-1} < 1$

Figure 7 shows plots for $E[W(i,i)]$ and $E[W(i,j)]$, the average wait times in the favorite memory case. As one would expect the wait time of a processor waiting for its favorite memory decreases to zero as $p \to 1$ (equation 29). However, the wait time of a processor waiting for a non-favorite memory grows unboundedly as $p \to 1$ (equation 36). Nevertheless, the overall average wait time experienced by a processor[3] is lower in the favorite memory case than in the uniform request case. This is in contrast to the results suggested by the approximate model describe in [SeD79] where the favorite memory of a processor is allowed to migrate.

All simulations were run with lengths of 100,000 system cycles the Appendix contains a tabular summary of the simulation data. From earlier experiments it was found that as r_i decreases the runs become less stable unless the number of simulation cycles is increased over 100,000. Thus $r_i = 0.1$ is about the minimum value used for our simulations.

Examination off the simulation results shows that for the most part $E[BW] > E[BW]_{sim}$. This effect was previously noted in [CKL77] for the case when $r=1$, and is caused by the simulator resubmitting blocked requests until they are finally accepted. As noted earlier, resubmission of blocked requests is more realistic than our model which operates under the IRP assumptions. Thus, we can consider our model as yielding slightly optimistic results. Notice that the above discrepancy essentially disappears when r becomes small.

[3] $\sum_{j=1}^{N} Pr\{A_{ij} \mid S_{ij}\} E[W(i,j)]$

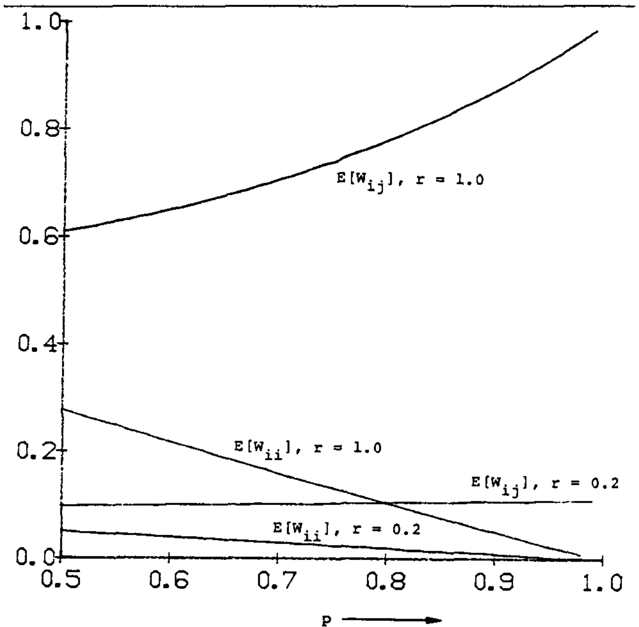


Figure 7.

5. Conclusion

In conclusion, the analysis presented above allows one to compute various performance figures for an N-M crossbar. In particular, in the case of each processor having a favorite memory, i.e., being more likely to request one particular memory than any other, new closed form solutions for the probability of request acceptance and the associated wait times were developed. The simulation results show that these performance figures are a close approximation to the actual behavior of the crossbar provided the request distributions are an accurate model of the request behavior of the processors.

6. References

[BaS76]
F. Baskett, and A. J. Smith, "Interference in Multiprocessor Computer Systems with Interleaved Memory," *CACM*, Vol. 19, No. 6, June 1976, pp. 327-334.

[Bat76]
K. E. Batcher, "The flip network in STARAN," *Proc. of the 1976 Int'l Conf. on Parallel Processing*, August 1976, pp. 65-71.

[Ben65]
V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York 1965.

[Bha75]
D. P. Bhandarkar, "Analysis of Memory Interference in Multiprocessors," *IEEE Trans. Computers*, Vol. C-24, No. 9, September 1975, pp. 897-908.

[BrD77]
F. A. Briggs, and E. S. Davidson, "Organization of Semiconductor Memories for Parallel-Pipelined

Processors," *IEEE Trans. Computers*, Vol. C-26, No. 2 February 1977, pp. 162-169.

[CKL77]
D. Chang, P.J. Kuck, and D. H Lawrie, "On the Effective Bandwidth of Parallel Memories," *IEEE Trans. Computers*, Vol. C-26, No. 5 May 1977, pp. 480-490.

[Clo53]
C. Clos, "A Study of Non-blocking Switching Networks," *The Bell System Technical Journal*, Vol. 32, March 1953, pp. 406-424.

[Fen74]
T. Feng, "Data Manipulating Functions in Parallel Processes and their Implementations," *IEEE Trans. Computers*, Vol. C-23, No. 3, March 1974, pp. 309-318.

[Fra80]
M. A. Franklin, "VLSI Performance Comparison of Banyan and Crossbar Connection Networks," in [Sie80], pp. 20-28.

[GoL73]
R. Goke, G. J. Lipovski, "Banyan Networks for Partitioning on Multiprocessor Systems," *Proc. of the 1-st Annual Symposium on Computer Architecture*, 1973, pp.21-30.

[Hoo77]
C. H. Hoogendoorn, "A General Model for Memory Interference in Multiprocessors," *IEEE Trans. Computers*, Vol. C-26, No. 10, October 1977, pp. 998-1005.

[Knu68]
D. E. Knuth, *The Art of Computer Programming Vol. 1: Fundamental Algorithms*, Addison Wesley, 1968.

[Law75]
D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Computers*, Vol. C-24, No. 18, December 1975, pp. 1145-1155.

[LiM]
G. J. Lipovski, M. Malek, "A Theory for Multicomputer Interconnection Networks" *IEEE Trans. Computers*, (to appear).

[MaM81a]
B. A. Makrucki, T. N. Mudge, *VLSI Design of a Crossbar Switch*, SEL Report No. 149, Department of Electrical and Computer Engineering, University of Michigan, January 1981.

[MaM81b]
B. A. Makrucki, T. N. Mudge, *Probabilistic Analysis of a Crossbar Switch*, SEL Report No. 150, Department of Electrical and Computer Engineering, University of Michigan, March 1981.

[Pat79]
J. H. Patel, "Processor-Memory Interconnections for Multiprocessors," *Proc. 6-th Annual Symposium on Computer Architecture*, IEEE, April 1979, pp. 166-177.

[Pat81]
J. H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. Computers*, Vol. C-30, No. 10, October 1981, pp. 771-780.

[Pea77]
M. C. Pease, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Trans. Computers*, Vol. C-26, No. 5, May 1977, pp. 458-473.

[Pip75]
N. Pippenger, "On Crossbar Switching Networks," *IEEE Trans. Communications*, Vol. COM-23, No. 6, June 1975, pp. 646-659.

[Rav72]
C. V. Ravi, "On the Bandwidth and Interference in Interleaved Memory Systems," *IEEE Trans. Computers*, Vol. C-21, No. 8, August 1972, pp. 899-901.

[Rau79]
B. R. Rau, "Interleaved Memory Bandwidth in a Model of a Multiprocessor Computer System," *IEEE Trans. Computers*, Vol. C-28, No. 9, September 1979, pp. 678-681.

[SeD79]
A, S. Sethi, N. Deo, "Interference in Multiprocessor Systems with Localized Memory Access Probabilities," *IEEE Trans. Computers*, Vol. C-28, No. 2, pp. 157-163.

[Sie80]
H. J. Siegel, (Ed.), *Proceedings of the Workshop on Interconnection Networks*, Purdue University, April 21-22, 1980.

[SiM81a]
H. J. Siegel, R. J. McMillen, "Using the Augmented Data Manipulator Network in PASM," *Computer*, Vol. 14, No. 2, February 1981, pp. 23-33.

[SiM81b]
H. J. Siegel, R. J. McMillen, "The Multistage Cube: A Versatile Interconnection Network," *Computer*, Vol. 14, No. 12, December 1981 (to appear).

[SiS78]
H. J. Siegel, S. D. Smith, "Study of Multistage SIMD Interconnection Networks" *Proc. of the %-th Annual Symposium on Computer Architecture*, April 1978, pp. 223-229.

[Str70]
W. D. Strecker, *Analysis of the Instruction Execution Rate in Certain Computer Structures*, Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, 1970.

[Tho80]
C. D. Thompson, *A Complexity Theory for VLSI*, Ph.D. Thesis, Carnegie-Mellon Univ., Computer Science Dept, August 1980.

[WuF80]
C-L. Wu, T-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers*, Vol. C-29, No. 8, August 1980, pp. 694-702.

7. Appendix

The following is a summary of simulation data for several system sizes and request rates/distributions.

UNIFORM-DISTRIBUTION-SIMULATIONS										
N	*M*	*r*	1−*pa*	*WF*	*E*[*W*]	*bw*1	*bw*2	*bw*3	*E*[*BW*]*sim*	*E*[*BW*]
4	4	1.0	.316	.34	.463	2.62	2.62	2.617	2.619	2.734
4	4	.5	.172	.20	.208	1.78	1.78	1.77	1.78	1.66
4	4	.1	.037	.035	.038	.40	.40	.40	.40	.39
8	8	1.0	.34	.38	.523	4.95	4.95	4.94	4.95	5.25
8	8	.05	.02	.40	.022	.40	.40	.39	.40	.39
4	8	1.0	.17	.183	.208	3.27	3.26	3.26	3.26	3.31
4	8	.5	.09	.1	.099	1.9	1.9	1.9	1.9	1.82
4	16	1.0	.09	.1	.099	3.62	3.63	3.63	3.63	3.64
4	16	.5	.05	.048	.048	1.93	1.93	1.93	1.93	1.91
8	4	1.0	.55	.59	1.22	3.26	3.26	3.26	3.26	3.6
8	4	.5	.34	.45	.523	2.83	2.83	2.83	2.83	2.63

where,
N - the number of processors
M - the number of memory banks
r - the processor request rate
pa - $Pr\{A_{ij} \mid S_{ij}\}$
WF - the simulation waiting fraction, it is the average over three runs
*bw*1 - the simulation bandwidth, for the first run
*bw*2 - the simulation bandwidth, for the second run
*bw*3 - the simulation bandwidth, for the third run
E[*BW*] - the computed average bandwidth
E[*BW*]*sim* - the average simulation bandwidth, it is the average of *bw*1, *bw*2, *bw*3
E[*BW*] - the computed average waiting time

FAVORITE-MEMORY-DISTRIBUTION-SIMULATIONS-I												
N	*M*	*r*	*p*	*pamax*	*pamin*	*wfmin*	*wfmax*	*bw*1	*bw*2	*bw*3	*E*[*BW*]	*E*[*BW*]*sim*
4	16	1.0	0.55	.96	.71	.05	.30	3.67	3.67	3.67	3.73	3.67
4	16	0.5	0.55	.98	.85	.022	.15	1.97	1.96	1.96	1.93	1.96
4	16	0.1	0.55	.99	.97	.005	.037	.40	.40	.40	.40	.40
4	16	1.0	0.40	.96	.78	.060	.235	3.64	3.64	3.65	3.68	3.64
4	16	0.5	0.40	.97	.88	.030	.12	1.95	1.96	1.96	1.92	1.96
4	16	0.1	0.40	.99	.98	.005	.025	.40	.40	.40	.40	.40
4	16	1.0	0.85	.99	.57	.018	.43	3.82	3.82	3.83	3.89	3.82
4	16	0.5	0.85	.99	.78	.008	.22	1.98	1.98	1.99	1.97	1.98
4	16	0.1	0.85	.99	.96	.01	.04	.40	.40	.40	.40	.40

These simulations were run using request distributions in which each processor requests a favorite memory with probability "*p*" and the remaining uniformly.

pamax - $Pr\{A_{ij} \mid S_{ij}\}$ for the preferred memory, and any memory preferred by no processor
pamin - $Pr\{A_{ij} \mid S_{ij}\}$ for any other memory
wfmin - *WF*, from the three simulations, for the preferred memory
wfmax - *WF*, from the three simulations, for any other memory

FAVORITE-MEMORY-DISTRIBUTION-SIMULATIONS-II												
N	*M*	*r*	*p*	*pamax*	*pamin*	*wfmin*	*wfmax*	*bw*1	*bw*2	*bw*3	*E*[*BW*]	*E*[*BW*]*sim*
8	4	1.0	.40	.48	.44	.55	.60	3.31	3.31	3.31	3.62	3.31
8	4	0.5	.40	.68	.65	.42	.46	2.85	2.85	2.85	2.64	2.85
8	4	0.1	.40	.92	.91	.09	.10	.79	.79	.79	.73	.79
8	4	1.0	.85	.51	.36	.49	.64	3.83	3.83	3.83	3.93	3.83
8	4	0.5	.85	.74	.60	.32	.47	3.14	3.15	3.14	2.86	3.14
8	4	0.1	.85	.94	.90	.06	.10	.80	.79	.79	.75	.79

Since this is an 8-4 connector, request "compression" must take place. In these distributions, two processors prefer a given memory (with magnitude "*p*") and their remaining probability distribution is uniform over all other memories. There are then three other pairs of processors in a similar situation. Thus each memory is preferred by two processors.