An Interconnection Network That Exploits Locality of Communication

Kenneth C. Sevcik
Computer Systems Research Institute
University of Toronto

Xiao-Nan Tan
Courant Institute
New York University

Abstract

Communication among processors in multicomputer systems is seldom perfectly balanced no matter how much care is taken in allocating the workload across the processors. Instead, the rates of communication between certain pairs of components are often much higher than the rates between other pairs. In some cases, the majority of communication reflects the structure of the underlying algorithm in the form of a ring, a mesh, or a binary tree, etc.

Some interconnection networks can exploit such asymmetries and locality by allowing more efficient communication between some pairs of processors than others. The combination routing scheme used with shuffle/exchange and exchange/shuffle networks is one such interconnection network.

In this paper, we will consider several patterns of structure and locality of communication among software components assigned to processors. In each case, we will identify a mapping between components and processors so that high volume communication paths correspond to connections supported most efficiently by the interconnection network. We show that locality of communication can have a profound effect on the efficiency of communication in a multicomputer.

Several types of multiprocessors and multicomputers being designed for applications in artificial intelligence are likely to exhibit locality of communication of a form suitable for exploitation by such interconnection networks.

1. Introduction

Many emerging large-scale applications of computers will require highly parallel systems to be successful. Two such applications are real-time computer vision and logical inference from knowledge bases. Consequently, in the Japanese fifth-generation computing project as well as in many related projects, there is an emphasis on the design of highly parallel systems that include hundreds or thousands of processors. Many distinct types of architectures are being investigated: dataflow [GuKW85], reduction [CrFR86], and inference [MuKO84, Hert86, Tana86], among others.

When a large number of functionally equivalent processing elements are available in a system, one aspect of attaining efficient execution of the application is to divide the computational load into a number of components so that it can be spread evenly (more or less) across the processors. This difficult task is currently left to the application programmer in most multiprocessing systems. Ideally, compilers will become better at partitioning application programs automatically in the future. Except for special applications in which there is a fortunate match between the number of processors and the problem size, however, it will be impossible for either programmers or compilers to simultaneously (1) balance the total work across all available processors (so that they are all essentially equally utilized), and (2) balance communication so that the rate between every pair of processors is about the same.

It is highly likely that instead there will be a high degree of locality of communication. The software components assigned to certain pairs of processors will tend to communicate much more frequently than other pairs. The natural tendency toward this kind of distribution is supported by the work of Zipf [Zipf49].

If locality of communication is inevitably present when applications are implemented on highly-parallel systems, it is a natural step to attempt to exploit it. The communication pattern among processors is determined by two major factors: (1) the natural communication structure of the algorithm (which might be a mesh in two or three dimensions, for example), and (2) the way in which software components of the application are mapped onto physical processing elements. For a specific application and mapping strategy, it is possible to estimate the relative communication rates between any two processors.

In highly-parallel systems, some form of interconnection network must provide a path for communication between pairs of processors. Some interconnection networks are "symmetric" in that communication is equally efficient between all pairs of processors. A standard omega network [Lawrie75] is an example of one such network. Other interconnection networks provide more efficient communication for certain processor pairs than for others. For example, in a hypercube interconnection [Seit85], a message from one processor to another may have to traverse anywhere from 1 to log₂N links (where N is the number of processors). Similarly, the
shuffle/exchange and exchange/shuffle network on which we focus in this paper allows packets sent from one processor to another to be delivered after anywhere from 1 to \( \log_2 N \) cycles through the interconnection network.

For a parallel processing system in which the interconnection network is not symmetric, an important step in efficiently implementing an application is the procedure of assigning software components to processors in a manner such that the higher volume communication paths between components correspond to the paths that are more efficiently supported by the interconnection network. This step is not always easy, as will be seen in later sections.

For our purposes, we will view the task of implementing an application on a parallel processor as including the following steps:

1. Partition the application into a number of software components that corresponds to the number of processors available.
2. Estimate the volume of communication required between each pair of components. (As we shall see in section 3, this information can be conceptually represented as an \( N \times N \) "communication" matrix for \( N \) software components.)
3. Characterize the "distance" matrix of the interconnection network through which the processors will communicate. (Again, this can be represented in an \( N \times N \) matrix, where the \((i, j)\)th entry is the relative cost or delay in sending a message from processor \( i \) to processor \( j \).
4. Map the \( N \) software components onto the \( N \) functionally equivalent processors, in such a manner that frequently used communication paths correspond to short paths in the interconnection network's distance matrix.

The remainder of the paper is structured as follows. Section 2 gives a survey of related prior work. Section 3 introduces a model of communication locality for applications implemented on parallel machines. Section 4 explores the use of combination routing in the shuffle/exchange and exchange/unshuffle interconnection network to exploit locality in the communication pattern of applications on parallel processors. Section 5 contains experimental results based on simulation that show the benefits of exploiting communication locality. Section 6 ends the paper with conclusions and discussion.

2. Background

A. Interconnection Networks

A great variety of interconnection strategies for parallel processors have been conceived and analyzed [Feng81, Bae876, Bene65]. Among these, we will focus primarily on those based on the shuffle/exchange interconnection pattern [Ston71]. A fundamental network of this class is the omega network [Lawr75]. An omega network of \( N \) processors consists of \( \log_2 N \) stages each of which involves \( N/2 \) \( 2 \times 2 \) switches (see figure 2.1). If we consider routing packets among \( N \) processors through an omega network, a packet from one processor can reach any other processor by taking the appropriate path through each switch it encounters in passing through the \( \log_2 N \) stages of the network.

Figure 2.1. An Omega network of \( \log_2 N \) shuffle exchange stages (for \( N = 8 \).)
When two or more packets are progressing through an omega network (or some variant) simultaneously, there is a chance that two packets will arrive at a particular 2 × 2 switch at the same time. In such a case, if they require different output paths from the switch, then they pass through simultaneously, not impeding one another. If they both require the same output path from the switch, however, then only one can proceed normally. The other is either delayed for one cycle in a buffer (if one is present) or is misrouted, causing it to have to be resent.

The literature on interconnection networks is vast. To limit the scope in relation to the material in this paper, we shall consider only

- packet routing through an interconnection network as opposed to permutation connections for establishing virtual circuits [LePV81, YeLa81],
- schemes in which the number of connections per component is constant at two rather than dependent on the number of processors (e.g., log₂ N connections per processor in hypercubes [Seitg85]),
- interconnection networks composed of 2 × 2 crossbar switches rather than crossbars of higher dimension,
- networks that are not enhanced or extended for increased reliability [ReKu84].

Even within these constraints, many variants of the omega network have been described in the literature. Some results have been obtained about the equivalence of certain classes of networks [WuFe80b, BeFJ85, OrOr85].

The omega network consists of log₂ N stages, each of which is based on the same shuffle/exchange connection pattern. Thus, it is possible to provide the same interconnection structure by using a single shuffle/exchange pattern with feedback. In this case, each packet cycles through the single-stage log₂ N times. The single-stage version of a shuffle-exchange (s/e) network has both hardware cost and throughput capacity reduced by a factor of log₂ N relative to a full omega network. Many properties of a single-stage shuffle/exchange network with buffering at each 2 × 2 switch are presented by Chen, et al. [CLYP81].

A packet originating at the source processor with address S = s₀ s₁ . . . sₙ₋₁ (where n = log₂ N) progresses toward destination processor D = d₀ d₁ . . . dₙ₋₁ dₙ through a shuffle/exchange interconnection network as follows. The shuffle effectively discards the high order bit of the address (since processors 0 sₙ₋₂ . . . s₁ s₀ and 1 sₙ₋₂ . . . s₁ s₀ are the two inputs to the same 2 × 2 switch). The exchange then effectively allows either a zero or one to be appended to the low order end of the address, depending on whether the packet takes the upper or lower exit from the switch. Overall, the passage through one stage shifts the address left by one position (causing the high order bit to be lost) and inserting a 0 or 1 on the right end. If routing at each switch is systematically chosen to be the next bit of the address of the destination processor, then the destination is reached after exactly log₂ N stages. (Figure 2.2 illustrates the sequence of intermediate addresses.)

A variation of the omega network that is particularly important to our work is the reverse-exchange network [WuFe80b]. This network can be viewed as an "exchange/unshuffle" network, which is essentially a mirror image of an omega network. We shall refer to the single-stage version of a reverse-exchange network as an exchange-unshuffle (e/u) network. The pattern of addresses traced out by a packet cycling through an exchange-unshuffle network is a sequence of addresses in which each one is obtained from the previous one by a right shift losing the low order bit (rather than a left shift) and an insertion of a high-order bit (rather than a low order bit) depending on the choice of path through the 2 × 2 switch. The close relationship between shuffle/exchange and exchange/unshuffle connections is further evidenced by the fact that the same internal structure can provide either routing pattern. In a standard shuffle exchange network, it is only necessary to reconnect the processors so that processor bₙ₋₁ bₙ₋₂ . . . b₁ b₀ is connected to input b₀ b₁ . . . bₙ₋₁ (i.e. bit-reversal of the address) in order to obtain the effect of reverse/exchange or exchange/unshuffle routing.

B. Combination Routing in s/e & e/u Networks

By considering the address transformation that occurs in shuffle/exchange (and exchange/unshuffle) networks, one observes that log₂ N cycles through a single-stage network are not always needed to route a packet from its source processor to its destination. In particular, if the last k bits of the source address match the first k bits of the destination address, then only n - k steps are required to deliver the packet. Once the match of the substrings is noted it is only necessary to route the packet through switches according to the last n - k bits of the destination address, rather than starting with the initial bits of the destination address. Yew [Yew81] noticed the possibility of this approach and proposed the "fast-finishing" routing algorithm for single-stage shuffle/exchange networks. Some experimental results for the "fast-finishing" technique are reported by Chen, et al. [CLYP81].

![Figure 2.2. The sequence of intermediate addresses encountered by a message moving from source S to destination D.](image-url)
We have previously proposed a generalization of fast-finishing routing for use with s/e & e/u interconnection networks, which are composed of one shuffle/exchange stage together with one exchange/unshuffle stage [TaSe87a]. Such a network is illustrated in figure 2.3. Our routing technique is called combination routing and it is based on exploiting common structure between source and destination addresses to the fullest extent possible in a s/e & e/u network. If a length $k$ substring of the source address is identical to a corresponding substring at offset $x$ in the destination address (i.e., $s_{kp-1} s_{kp-2} \ldots s_{kp} s_{p} \equiv d_{kp+p-1} d_{kp+p-2} \ldots d_{p+k+1} d_{p+k}$) then the packet can be routed from $S$ to $D$ in $\min(n, 2(n-k)-1-x)$ steps. Yew's fast-finishing routing (which we have called "one-way" routing) is a special case of combination routing where $p=0$ and $x=n-k$, and all routing occurs through the shuffle/exchange part of the s/e & e/u network. Another special case occurs when a length $k$ prefix of the source address is identical to the length $k$ suffix of the destination address (so that $p=n-k$ and $x=-(n-k)$). In this case, all routing occurs in the exchange/unshuffle part of the s/e & e/u network.

In an earlier paper [TaSe87a], we used Monte Carlo simulation to compare the performance of combination routing in a s/e & e/u network with standard routing in a single-stage shuffle-exchange network. In the case where communication patterns are symmetric among all processors (i.e., messages from one processor are equally likely to be destined for each other processor), the simulation results showed that combination routing in the s/e & e/u network provides significantly better performance (higher capacity and lower delivery times) than standard routing in a single-stage shuffle-exchange network. In this paper, we shall treat the case in which the communication patterns among processors contain various forms of asymmetry, and we will find that the benefits of combination routing relative to standard routing are magnified when such localities of communication are present.

The overall significance of locality of communication in assessing the performance of interconnection networks is discussed by Reed and Grunwald [ReGr87]. Other authors have considered specific forms of locality and have proposed changes or extensions to the design of interconnection net-

Figure 2.3. An s/e & e/u interconnection network constructed from one shuffle exchange interconnection and one exchange unshuffle interconnection.
works to handle such locality. Bhuyan [Bhuy85] compared a crossbar switch and a multistage interconnection network with respect to their performance when each processor has a "favorite" memory that it accesses more frequently than the others. He found that as the locality (degree of favoritism) increased, the delays encountered due to congestion in the interconnection network decreased.

A different type of locality occurs when all (or several) processors favor access to the same memory. This creates a "hot-spot" in the interconnection network, and the associated problem of tree saturation. Several techniques for dealing with hot-spots have been proposed. One is to simply increase the amount of hardware in the interconnection network to provide alternate paths through the hot-spot. A second technique is "combining", in which messages destined for the same memory are detected and combined in order to reduce the number of distinct messages that must pass through the hot-spot. In the IBM RP3 and the NYU Ultracomputer, combining is done by special hardware in the interconnection network [PhNo85]. More recently, Yew, Tzeng and Lawrie have suggested that the combining can be done nearly as effectively and at significant lower cost in software [YeTL87]. Finally, Nichols and Messerschmitt have proposed an approach in which the connections of the network are chosen specifically to suit a known pattern of communication traffic [NiMe87]. This approach requires that the pattern of communication be both known precisely and stable over a long period of time.

3. A Model Of Communication Locality

A. Communication Matrices

In designing an application to run on a multiprocessor, the software must be partitioned into components that will execute on different processors. This partitioning, which may be done by either a compiler or by an application designer, is intended to balance the computational load across the available processors. Once the partitioning is done, it is possible to estimate the amount of communication required between the components assigned to the various processors. This information can be represented in a communication matrix. If \( N \) is the number of components into which the software is divided, the communication matrix is an \( N \times N \) matrix whose \((i,j)\)th entry represents the rate (either absolute or relative) of communication between component \(i\) and component \(j\).

B. A One-Parameter Model of Locality

Various multiprocessor applications exhibit varying degrees of locality in communication. It is impractical to investigate a large variety of arbitrary patterns of communication locality. For this reason, we study here a parameterized model of communication. As the single parameter is varied, the pattern of communication ranges from uniform and symmetric to highly localized.

Figure 3.1 illustrates the structure of our model of locality. Each component is a member of a sequence of \( n = \log_2 N \) neighborhoods of size \( 2^l \) through \( 2^n \) respectively. Each component communicates at relative rate \( x^j \) with each of the \( 2^{l-1} \) other components that are in the same level \( j \) neighborhood (but not in any common smaller neighborhood). The parameter \( x \) lies in the range 0 to 1. When it is one, communication rates are symmetric and uniform among all components.

When the value of \( x \) is less than one, the rate of communication to each component in one neighborhood is smaller by the factor \( x \) than the rates to components in the next smaller neighborhood. As \( x \) approaches zero, each component tends to communicate only with the one other component in its smallest neighborhood (its "buddy"). The model of locality that we propose here is a multi-level generalization of the locality model suggested by Reed and Grunwald [ReGr87].

There are two important features to note in this model of locality. First, each component is allowed to communicate with any other component (although the rate of communication may be arbitrarily low). Second, as the single parameter is varied between zero and one, the model exhibits locality of communication ranging from no locality at all to extreme locality.

C. Some Specific Common Structures of Locality

Certain applications, by their structure, lead naturally to specific patterns of communication locality. In this section, we mention three of these. Many multiprocessor applications, particularly when the number of processors is rather small, can be arranged so that all communication is with a component's two neighbors when the components are arranged in a ring structure. A number of numerical applications, for example for solving systems of linear equations or partial differential equations [OkVo85], etc., have been implemented with a logical communication patterns based on a ring. Figure 3.2(a) illus-
trates the ring interconnection, and figure 3.3(a) shows the corresponding communication matrix (for the case n=16). A common variation on this communication pattern involves a daisy-chained ring in which each processor communicates directly with the next two processors along the ring in each direction.

Multiprocessor applications that have components of software corresponding to portions of two-dimensional or three-dimensional space lead naturally to mesh (or grid or toroid) structures in two or three dimensions. For example, some computational fluid dynamics applications (for weather forecasting or aerodynamic structure design) involve the simu-

![Figure 3.3](image)

Figure 3.3. Communication matrices that correspond to the specific common structures of locality shown in figure 3.2.

![Figure 3.2](image)

Figure 3.2. Some specific common structures of locality.
lation of air flow in two or three dimensional space [OrVo85]. In these applications, the software components corresponding to one area in space communicate with those corresponding to adjacent areas in space. Figure 3.2(b) illustrates a mesh connection in two-dimensional space. Figure 3.3(b) shows the corresponding communication matrix.

A third common structure of communication is that of a binary tree. Tree structures can arise when each piece of software uses a "divide and conquer approach" in which it does some work, but also creates units of work for other processors. Some recent designs of inference machines have been based on this strategy [Tan86]. Figure 3.2(c) illustrates a binary tree structure interconnection among sixteen processors. Figure 3.3(c) shows the corresponding communication matrix.

If the communication among processors for a specific application takes the form of some special structure such as the three discussed in this section, then a multiprocessor built with that specific hardware interconnection structure is clearly the most suitable interconnection structure for that particular application. There are two justifications, however, for considering such applications. First, while a significant portion of the required communication may follow the specific structure, it is seldom the case that there is not also the need for some amount of more general communication among the processors. In such situations, it is important that both types of communication be efficiently supported. Second, most large multiprocessors will not be devoted to a single application with a specific communication structure. Instead, they will be required to support a variety of applications with differing underlying communication structures.

In the following sections, we will investigate the problem of using a single interconnection network among a number of processors to support various applications with diverse communication structures, while taking advantage of whatever communication locality exists in the application. We shall show that the s/e & e/u network with combination routing can exploit locality to yield more efficient communication than is possible with other interconnection networks based on 2 x 2 switches and their routing algorithms.

4. Using Combination Routing to Exploit Locality of Communication

The communication matrix of an application indicates the pattern of communication independent of any particular interconnection structure of the multiprocessor upon which the application is to run. In this section, we investigate the impact of supporting an application with an associated communication matrix on a multiprocessor with a specific interconnection structure.

In a multiprocessor system with a given interconnection network, the cost (usually measured in time) of communicating from one processor to another can be represented in a two-dimensional distance matrix. The (i,j)th entry of the matrix indicates the relative cost in communication delay incurred when processor i sends a message to processor j. Figure 4.1 shows representative distance matrices for several common interconnection structures in current use in multiprocessor systems. Several properties of these distance matrices may be noted. If all elements (off the main diagonal) are identical, as for the bus interconnection, then the communication distance is the same for each processor pair. If the cost matrix is symmetric about the main diagonal, then for each pair of processors, the cost of communicating is the same in either direction, and we consider the communication distances to be symmetric. The communication distances are symmetric for the bi-directional ring and the hypercube, but not for the unidirectional ring and one-way (or "fast finishing") [Yew81] routing.

When using an interconnection scheme with uniform distance, once the application is divided into components to be executed on each processor, there is no further issue that influences the overall efficiency of execution of the application. When the distance is non-uniform, however, the greatest efficiency in execution of the application is achieved if the smaller entries in the distance matrix of the interconnection network are made to match with the larger entries in the application’s communication matrix. The order of the software components in constructing the communication matrix was arbitrary. Thus, for a given application and a given interconnection matrix, we may seek a permutation of the rows and columns of the communication matrix that will minimize the overall communication cost of the application when executed on a multiprocessor with the given interconnection network.

More formally, we define $C$ and $D$ to be the communication and distance matrices respectively. Then we seek the function $p$ that is a permutation function on the integers 0 to $N-1$ and minimizes the sum of the products of the corresponding elements of $C$ and $D$:

$$\text{minimize } \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(p(i), p(j)) \times D(i,j)$$

with respect to permutation function $p$.

In general, this is a difficult combinatorial problem. It would become still more difficult if we were to take into account the fact that the communication between one pair of processors can degrade the efficiency of communication between another pair if the two paths intersect at some switch element. We assume that this effect is of secondary importance and we ignore it.

In the next two subsections, we shall discuss the matching problem above for some specific combinations of communication and distance matrices. First, we treat the communication matrix corresponding to our one-parameter model of locality, then we treat ring and tree type communication matrices. In all cases, we consider the distance matrix that corresponds to combination routing in a s/e & e/u interconnection network.

A. A One-Parameter Model of Locality

The way we constructed our one-parameter model of locality, the groups of components that intercommunicate are those whose addresses differ only in the low order bits. That is, localities of size $2^k$ consist of components whose addresses have the same first $n-k$ bits. Combination routing in s/e & e/u networks takes advantage of long common substrings in source
and destination address pairs. Thus, using the identity permutation to associate software components with processors leads to a reasonable matching of the communication and distance matrices.

For specific values of the address length, $n$, and the degree of locality, $x$, permutation functions better than the identity can be found. We have not, however, found an efficient algorithm for identifying the optimal permutation mapping of component addresses to processor addresses.

A modest improvement to the identity permutation over a wide range of $(n,x)$ values is obtained by a mapping that causes addresses within a locality to differ by a few bits at either end rather than always at the low order end. Let

$$p^*(b_{n-1} b_{n-2} b_2 b_1 b_0) = [b_{n-1} \oplus b_0] b_{n-1} b_{n-2} b_2 b_1$$

where $\oplus$ denotes exclusive or, with inverse mapping

$$p^{-1}(b_{n-1} b_{n-2} b_2 b_1 b_0) = b_{n-2} b_2 b_1 b_0 [b_{n-1} \oplus b_{n-2}]$$

Figure 4.2 illustrates the advantage of using permutation $p^*$ rather than the identity permutation. Figure 4.2(a) shows the routing distances within neighborhoods of sizes 2, 4, and 8 assuming that $n$ is large. Under the identity mapping, the relative cost of communication among the eight components is

$$1(2) + 2(4)x + 4(6)x^2 = 2 + 8x + 24x^2$$

After applying permutation $p^*$ to the eight addresses, the distances are reduced as shown in Figure 4.2(b). The relative cost is reduced to

$$1(2) + (2 + 4)x + (4 + 6 + 4 + 6)x^2 = 2 + 6x + 20x^2$$

The experimental results shown in the next section are based on the use of permutation $p^*$ in mapping software component addresses to processor addresses.

B. Special Types of Locality

In this section, we shall indicate appropriate permutation mappings for communication matrices reflecting two special locality patterns: ring and binary tree. Our assumption will be that any software component may communicate with any other, but that the majority of the communication in the application follows the special locality structure.

B.1 Ring Communication

Figure 3.3(a) showed a communication matrix in which each component communicates only with its immediate neighbors around a logical ring. In the logical ring on which the figure is based, the neighbors of address $A$ are addresses $(A + N - 1) \ mod \ N$ and $(A + 1) \ mod \ N$.

The identity permutation is an effective way of mapping this ring-based communication matrix onto a combination routing s/e & e/u network distance matrix. Taking this approach, the average communication distance is bounded above by the expression

$$\frac{\sum_{j=1}^{n/2} j}{2^{n-1}} + \frac{n}{2^{n/2}}$$

which approaches 4 asymptotically from below as $n$ grows. Figure 4.3 shows how this routing distance compares with

---

**Figure 4.1.** Distance matrices for some common interconnection structures (for the case of $N = 8$ processors).
Average routing distance with the identity mapping:

\[ 1 \times (2) + x (2(4)) + x^2 (4(6)) = 2 + 8x + 24x^2 \]

(a). For the identity mapping.

| x----x000 | 0 | 2 | 4 | 4 | 6 | 6 | 6 |
| x----x001 | 2 | 0 | 4 | 4 | 6 | 6 | 6 |
| x----x010 | 4 | 4 | 0 | 2 | 6 | 6 | 6 |
| x----x011 | 4 | 4 | 2 | 0 | 6 | 6 | 6 |
| x----x100 | 6 | 6 | 6 | 2 | 0 | 4 | 4 |
| x----x101 | 6 | 6 | 6 | 6 | 2 | 0 | 4 |
| x----x110 | 6 | 6 | 6 | 6 | 4 | 0 | 2 |
| x----x111 | 6 | 6 | 6 | 6 | 4 | 4 | 2 |

Average routing distance with the mapping, \( p^* \):

\[ 1 \times (2) + x (2(4)) + x^2 (4(6)) = 2 + 6x + 20x^2 \]

(b). For the mapping, \( p^* \).

Figure 4.2. Improvement of mapping \( p^* \) over the identity mapping for the one-parameter model of locality.

5. Experimental Results

In order to evaluate the relative performance of combination routing and standard shuffle exchange in the presence of contention among messages, we carried out simulation experiments for both the one-parameter model of locality and for the binary tree based communication pattern.

In each of the simulations, each processor started with a single message. The destination of that message was chosen according to the model of communication locality under investigation. Cycles of the network were simulated until all messages were successfully delivered, and the distribution of delivery times was recorded. Each 2 \times 2 switch was assumed to have buffer size of 2. A sufficient number of trials were run to give 95% confidence that the errors in the mean values were less than 2%.

A. The One-Parameter Model of Locality

Trials were done for from \( 2^4 \) to \( 2^{12} \) nodes and degrees of locality \( x=1, .5, .25, .125, \) and .0625. The results are shown in figure 5.1 and table 5.1. Both the expected average delivery time and the expected maximum delivery time become smaller as the locality of communication becomes stronger (which happens as \( x \) gets smaller). Also, note that the effect becomes more pronounced for larger numbers of processors. With locality parameter \( x=.0625 \) and \( N=4096 \) nodes, the average delivery time is about 4.4 and the expected maximum delivery time about 11. With standard shuffle exchange routing, on the other hand, these numbers are about 18 and 50 respectively. Under standard shuffle exchange routing, we found that changing the locality parameter had no significant effect on the average or expected maximum routing distance.

B.2 A Binary Tree Structure

Figure 3.3(c) showed a communication matrix in which components communicate according to a binary tree structure. In particular, node \( j \) has parent \( \lfloor j/2 \rfloor \) and children \( 2j \) and \( 2j+1 \). (This is the same addressing structure used to linearize a binary tree in heapsort, for example.) In this case, simply using the identity mapping leads to an optimal routing distance of one by mapping component addresses to processor addresses according to a Gray code so that adjacent addresses on the logical ring differ by only a single bit.

5.2 A Binary Tree Locality Model

In the case of the binary tree communication model, we assumed that fraction \( f \) of all messages follow the binary tree pattern, while fraction \( 1-f \) is routed uniformly across all nodes. Of the messages sent on the binary tree, we tried two ways of
6.62
1.125
2.05
2.71
3.55
4.09
4.16
4.3 1
4.37
4.37
4.38

(a). Average delivery times

(b). Expected maximum delivery times.

Table 5.1. Average and expected maximum routing distance under the one-parameter model of locality for various degrees of locality.

weighting the selection: (1) \( \frac{1}{2} \) to the parent and to each child, and (2) \( \frac{1}{2} \) to the parent and \( \frac{1}{4} \) to each child.

The results of the experiments are shown in figure 5.2 and table 5.2. As the fraction of messages sent according to binary tree locality increases, both the average and maximum delivery times decrease sharply. There is little difference in performance between the \((\frac{1}{2}, \frac{1}{2}, \frac{1}{2})\) and \((\frac{1}{2}, \frac{1}{4}, \frac{1}{4})\) weightings.

6. Conclusions

Large-scale multiprocessors that are currently under development are intended to overcome the "von Neumann bottleneck" by supporting internal concurrency within an application program. This necessitates communication among the components of the application program that are executing on distinct processors. The efficiency of the interconnection network that supports this communication is a significant factor in determining the overall throughput and response times for applications. First, the communication component of overall response time may be reduced substantially by an effective interconnection network. Second, and perhaps more importantly, if each message is delivered in fewer interconnection network cycles, then the network's capacity is effectively increased.

In current systems, processing and synchronization delays are the largest components of overall response times, and communication delay is less significant. However, the number of processors in a multiprocessor is increasing, and we are developing more sophisticated techniques for decomposing applications into smaller units while keeping synchronization delays tolerable. These trends will make the efficiency of interconnection networks more and more critical to overall performance in the future.

In this paper, we have proposed a model of locality of communication in multiprocessor environments. A parameter permits the model to represent a full range of degrees of locality from none to total. In the case of combination routing in a s/e & e/u network, we have shown that the efficiency of communication increases substantially as the degree of locality of communication goes up. While our model represents a particular, highly structured form of locality, we would expect comparable gains in efficiency from many situations where communication is most frequent between pairs of processors whose processor sequence numbers are close to each other. This situation is likely to be encountered more and more as the number of processors in multiprocessor architectures increases. This is true because processor allocation will be done hierarchically, so that a range of processors (ordered by id number) will often be allocated to a cluster of interacting processors.
## Average and Maximum Delivery Times

<table>
<thead>
<tr>
<th>f</th>
<th>0.0</th>
<th>5</th>
<th>.75</th>
<th>.875</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.33</td>
<td>9.89</td>
<td>5.02</td>
<td>2.93</td>
<td>1.90</td>
<td>1.00</td>
</tr>
<tr>
<td>.50</td>
<td>9.89</td>
<td>5.02</td>
<td>2.93</td>
<td>1.90</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(a). Average delivery times (for \( n = 12 \))

<table>
<thead>
<tr>
<th>f</th>
<th>0.0</th>
<th>5</th>
<th>.75</th>
<th>.875</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.33</td>
<td>26.64</td>
<td>18.48</td>
<td>13.79</td>
<td>12.73</td>
<td>1.00</td>
</tr>
<tr>
<td>.50</td>
<td>26.64</td>
<td>18.36</td>
<td>13.83</td>
<td>12.74</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(b). Expected maximum delivery times (for \( n = 12 \)).

Table 5.2. Average and expected maximum routing distance under the binary tree model of locality.

Furthermore, if such multiprocessors are shared by independent applications, as is often necessitated by their expense, then it is likely that groups of processors with contiguous addresses will be allocated to a single application. So locality of communication in multiprocessors will tend to be stronger in the future.

We also explored two specific frequently encountered communication structures: binary trees and rings. For the binary tree structure, we showed that, with appropriate numbering of processors, all messages can be delivered in a single step in an s/e & e/u network. For the ring structure, the average number of steps required to deliver a message is at most four, no matter how large \( n \) becomes. In each case, \( \log_2 N \) steps would be required in an omega network. Finally, we used Monte Carlo simulation to quantify the gains to be expected in using an s/e & e/u network rather than a more conventional shuffle exchange type interconnection network. For all network sizes, the gain in performance increases significantly with the degree of locality of communication.

### References

[Batch76]

[Benes65]

[BeFr85]

[BhAg82]

[Bhuy85]

[CLYP11]

[CoFR86]

[Feng81]

[GGKMR83]

[GouKW85]

[Hert86]

[HuT86]

[Lawr75]

[LeKK86]
[LePV81]

[MuKO84]

[NIMe87]

[OrOr85]

[OrVo85]

[PiNo85]

[ReGr87]

[ReKu84]

[Schw80]

[Seit85]

[Sion71]

[Tan86]

[TeSe87a]

[TeSe87b]

[WuFe80b]

[WuFe81]

[Yew81]

[YeLa81]

[YeTL87]

[Zipf49]
G.K. Zipf, Human behavior and the principle of least effort, Addison-Wesley, 1949.