On Shortest Path Routing in Single Stage Shuffle-Exchange Networks*

Sunil Kim
Center for Supercomputing R & D
University of Illinois
1308 W. Main St.
Urbana, IL 61801-1351
kim@csrd.uiuc.edu

Alexander V. Veidenbaum
Department of EECS
University of Illinois
851 S. Morgan St.
Chicago, IL 60607-7053
alexv@eecs.uic.edu

Abstract

In this paper, we study routing in shuffle-exchange networks. Shuffle-exchange networks can have two different structures: multistage and single stage. Routing in multistage networks with $K \times K$ crossbar switches needs $\lceil \log_K N \rceil$ stage traversals for the connectivity between N inputs and N outputs. In single stage networks, less than $\lceil \log_K N \rceil$ traversals may be required depending on source and destination. We establish a theorem for routing from an input terminal to an output terminal at *any* stage in multistage networks. In the theorem, system size is limited only as a *multiple* of a crossbar switch size. This condition allows more flexible increments in system size. Based on the theorem, we derive an algorithm that generates routing tags for shortest path routing in single stage networks. We study the impact of shortest path routing on average internode distance, and by using trace-driven simulation, we evaluate its effect on shared memory systems. Our results show that the shortest path routing scheme, without much additional hardware, can improve the performance of shared memory systems by more than 30 % depending on system size and switch size.

1 Introduction

In multiprocessor systems, processors and memories are connected through interconnection networks. Communication and data sharing between these components are performed by sending and receiving messages which travel through interconnection networks. As the size of a system is increased to obtain greater computing power, the impact of interconnection networks on system performance and cost becomes more significant.

Among various interconnection network architectures, shuffle-exchange networks [2, 10, 13, 14] have been one of the most popular architectures. The networks have distributed self-routing capabilities, and when network size is increased, the internode distance and the number of switch components are increased logarithmically. These features make shuffle-exchange networks suitable for large-scale multiprocessor systems.

The shuffle-exchange networks can take two major configurations: multistage and single stage networks. The multistage network shown in Fig 1 consists of $\lceil \log_K N \rceil$ shuffle-exchange stages, where N is a network size, and K is a (crossbar) switch size. The crossbar switch is a basic building block of the network. The single stage network consists of only one stage that has a circular connection between its input and output terminals as shown in Figure 2.

Both types of networks use a common routing algorithm. For a network size equal to a power of the switch size, the algorithm uses a destination address as a routing tag without modification. Although this simple routing tag generation is an attractive feature of shuffle-exchange networks, the network size restriction it imposes is a serious limitation on incremental scalability of networks. When we want to increase a network size, the new size has to be K times larger.

In [2], a new approach is taken to allow a more flexible network size which can be a composite number instead of a power of K. In this approach, a network size needs to be factored into r numbers so that each terminal number of a network can be expressed as an r-tuple. The switch size at each stage corresponds to one of r numbers, implying that it may need a different switch size depending upon how the network size is factored. Still, the increased network size needs to be a multiple of the previous network size.

Padmanabhan [15] lifted the limitation of the network size from shuffle-exchange network routing, allowing a more flexible increase in network size. In his papers [14, 15], he laid out a theoretical framework for a routing scheme requiring only that the network size be a multiple of K and investigated the properties of the scheme. Based on this work, a routing tag can be generated from source and destination addresses and all possible multiple paths can be detected. In his routing scheme, which we will refer to as *multistage routing*, $\lceil \log_K N \rceil$ hops are required to reach a destination for both multistage and single stage networks. One hop is equal to one stage traversal.

In a single stage network, a node is connected to a input terminal of a switch that directly leads to other neighboring nodes whose input terminal is connected to the switch. A message needs to traverse a shuffle-exchange stage(a switch and a link) $\lceil \log_K N \rceil$ or less times to reach its destination. The direct connection structure and recirculation of messages allow a different routing algorithm in single stage net-

*This work was supported by NASA Ames Research Center under Grant No. NASA NCC 2-559, the Department of Energy under Grant No. DE-FG02-85ER25001, the National Science Foundation under Grant No. MIP-8920891, and University of Illinois.

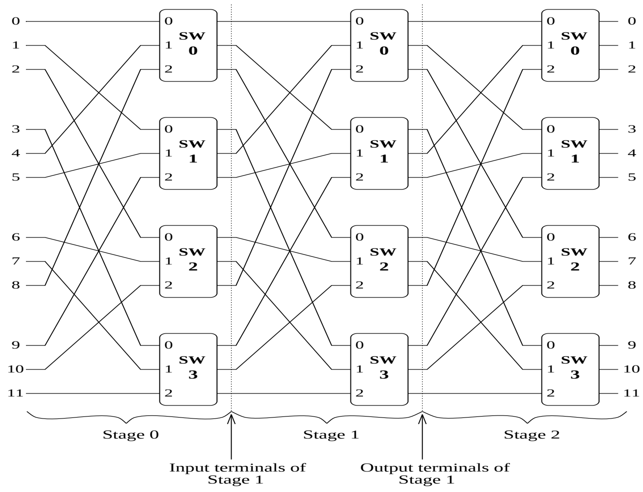

Figure 1: Multistage Shuffle-Exchange Network

works than in multistage networks, which provides a shortest path between nodes. Shortest path routing in single stage networks is important because it reduces internode distances and eliminates cycles in a message path. If a cycle is allowed in a message path, it creates deadlock situations, and a message collide with itself to complicate conflict resolution schemes. In this study, we develop an algorithm that finds a routing tag for a shortest path in single stage networks.

Although a *fast-finishing* routing algorithm was proposed in [17, 19, 20] to find shorter routes in single stage networks, it is limited to network sizes which are a power of K. In addition it has not been shown whether such a method guarantees a shortest path. The algorithm compare the destination and source addresses to find a matching pattern in both addresses. Suppose that the destination address is $M_n M_{n-1} \ldots M_0 T_m \ldots T_0$ and the source address is $S_m \ldots S_0 M_n M_{n-1} \ldots M_0$ such that there is a matching pattern $M_n M_{n-1} \ldots M_0$. Then, the unmatched address digits $T_m \ldots T_0$ are used for routing messages. Therefore, only a number of hops equal to the number of the unmatched address digits is necessary as opposed to $\log_K N$ hops.

In this paper, we extend and generalize Padmanabhan's work to provide a theoretical background for shortest path routing in single stage networks. The network size can be any multiple of K. An algorithm will be presented for the routing tag generation. We will evaluate the performance improvement achieved with the shortest path routing algorithm over the *multistage routing* algorithm in single stage networks.

2 Routing in Shuffle-Exchange Networks

First, we introduce a theorem for routing in multistage networks. Routing in single stage networks can be considered a special case of the theorem. Based on the theorem, we will develop an algorithm to find a routing tag for a shortest path in single stage networks. Later, we extend the theorem for the shortest path routing in *generalized de Bruijn Digraphs* [5], which is another variant of single stage network architectures.

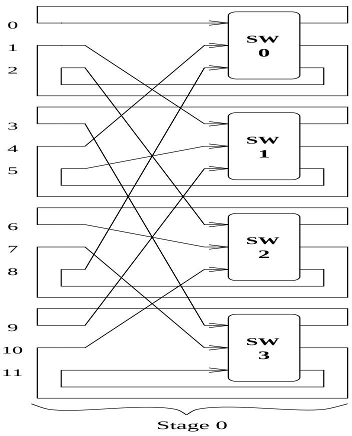

Figure 2: Single Stage Shuffle-Exchange Network

2.1 Network Structure

A K-ary shuffle-exchange multistage network provides connections between N input terminals and N output terminals. The network consists of several shuffle-exchange stages. Each stage has N input and N output terminals that are connected to N output terminals of the previous stage and N input terminals of the next stage, respectively, as shown in Figure 1. In a stage, there are N/K crossbar switches of size K(K inputs and K outputs). The N input terminals of a stage are connected to input ports of the switches through N links. The N link connections form a K-ary shuffle pattern.

Messages are injected at the input terminals of a network which are the input terminals of the first stage. They are delivered to the output terminals of the network which are the output terminals of the last stage. As it will be shown in the following theorems and corollaries, $\lceil \log_K N \rceil$ stages(hops) are needed for any output terminal of a network to be reachable from any input terminal of the network.

A single stage network consists of only one stage. The output terminals of the stage are reconnected to the input terminals of the stage as shown in Figure 2. Messages are injected at the input terminals of the network and circulate in the network to reach their destinations. The output terminals can be reached in $\lceil \log_K N \rceil$ hops or less because the input and output terminals are directly connected through switches.

First, we will define the shuffle connection in a stage using a definition from Padmanabhan [14, 15].

Definition 1 *Let N be a multiple of K. Then, the K-ary shuffle of N terminals is given by*

$$\pi(i) = (Ki + \lfloor Ki/N \rfloor) \bmod N, \quad 0 \leq i < N.$$

where terminal i maps to terminal $\pi(i)$.

In a stage of a shuffle-exchange network, input terminal i of a stage is connected to terminal $\pi(i)$ inside the stage through a link, and the terminal is connected to a corresponding input port of a switch in the stage. A message starting from terminal i reaches a terminal with address $\pi(i)$ after moving through the link of the shuffle connection.

The exchange routing function is performed at a crossbar switch where a routing tag T is used to select a requested output port. A switch takes the most significant digit from the routing tag T in the radix-K number representation and uses it to select an output port of the switch. The routing tag T is shifted by one digit to eliminate the most significant digit.

For a multistage network where the number of stages is fixed by its architecture, the tag T has the same number of digits as the number of stages. For a single stage network, the number of traversals of the network can vary depending on the source and destination, and thus vary the number of digits in T.

2.2 Multistage Shuffle-Exchange Networks

Routing in multistage networks differs from that in single stage networks. In this section, a routing theorem for multistage networks will be introduced based on the definitions in the previous section. The theorem provides a theoretical background not only for multistage routing in multistage networks but also for a shortest path routing in single stage networks. In addition, we will investigate a fault-tolerance property of multistage networks by extending the theorem.

We will start with a few Lemmas.

Lemma 1 *Let N be a multiple of K. The K-ary shuffle permutation can be specified by*

$$\pi(i) = Ki \bmod N + \lfloor Ki/N \rfloor, \quad 0 \le i < N.$$

The $Ki \bmod N$ is a multiple of K, and $\lfloor Ki/N \rfloor$ is the least significant digit of $\pi(i)$ in a radix-K number representation.

Proof: Let $m = Ki \bmod N$ and $\lfloor Ki/N \rfloor = t$, that is, $Ki = CN + m$, where $C \ge 0$, and $0 \le m < N$. Since $0 \le i < N$, $0 \le \lfloor Ki/N \rfloor = t < K$.
From Definition 1,

$$\begin{aligned} \pi(i) &= (Ki + \lfloor Ki/N \rfloor) \bmod N \\ &= (CN + m + t) \bmod N \\ &= (m + t) \bmod N. \end{aligned}$$

We will show that $m+t < N$. Since Ki and N are multiples of K, $m (= Ki - CN)$ is also a multiple of K, that is, $Ki \bmod N$ is a multiple of K.
Because both N and m are multiples of K and $m < N$, $K \le N - m$. Now, we have $t < K \le N - m$, and therefore $m + t < N$.
Since $m + t < N$, we have

$$\begin{aligned} \pi(i) &= (m + t) \bmod N \\ &= m + t \\ &= Ki \bmod N + \lfloor Ki/N \rfloor. \end{aligned}$$

Let $\pi(i)$ be represented in the radix-K number system, i.e. $\pi(i) = (m_n, m_{n-1}, \ldots, m_0)_K$.
From the above, $\lfloor Ki/N \rfloor < K$, and $Ki \bmod N$ is a multiple of K, i.e, $Ki \bmod N = (m_x, m_{x-1}, \ldots, 0)_K$. Thus, $\lfloor Ki/N \rfloor = m_0$ and $m_x = m_n, m_{x-1} = m_{n-1} \ldots$
Therefore, $\lfloor Ki/N \rfloor$ is the least significant digit of $\pi(i)$ in the radix-K number representation. □

In the above Lemma, we gave a new function describing the shuffle permutation, and examined how each term is related to the radix-K representation of $\pi(i)$. Note that $(Ki \bmod N)/K$ and $\lfloor Ki/N \rfloor$ correspond to a switch number in the stage and a input port number in a switch, respectively as shown in Figure 1.

Lemma 2 *A message starting at input terminal i of a stage and given an one-digit routing tag t_0 reaches output terminal $SX(i, t_0)$ of the stage specified by*

$$SX(i, t_0) = Ki \bmod N + t_0 = (Ki + t_0) \bmod N.$$

Proof: After a K-ary shuffle, terminal i maps to terminal $(m_n, m_{n-1}, \ldots, m_1, m_0)_K$, where $Ki \bmod N = (m_n, m_{n-1}, \ldots, m_1, 0)_K$ and $m_0 = \lfloor Ki/N \rfloor$. The routing tag t_0 causes the switch to route to its output port t_0 which thus replaces the least significant digit of $\pi(i)$, m_0 by t_0. Therefore, the output terminal after the shuffle and exchange routing in a stage is

$$SX(i, t_0) = Ki \bmod N + t_0.$$

Note that N and $Ki \bmod N$ are multiples of K, and $Ki \bmod N < N$. Since $t_0 < K$, $Ki \bmod N + t_0 < N$.
By applying the derivation used in Lemma 1, we have $Ki \bmod N + t_0 = (Ki + t_0) \bmod N$. □

The above Lemma defines the composite function of the shuffle and exchange routing in a stage causing a message to advance through a stage. As one can observe, the routing tag t_0 is added to Ki before the modulo N function is applied. This leads to the next Lemma generalizing the composite function.

Lemma 3 *After n stage traversals(n K-ary shuffle and exchange routing steps) with a routing tag $T = (t_{n-1}, \ldots, t_0)$, an output terminal $SX^n(i, T)$ in stage n can be reached from terminal i, where*

$$SX^n(i, T) = (K^n i + K^{n-1} t_{n-1} + \ldots + K t_1 + t_0) \bmod N.$$

Proof: From the previous Lemma, we proved $SX^1(i, T) = (Ki + t_{n-1}) \bmod N$ where $T = (t_{n-1})$. By the following the derivation for the K-ary shuffle and exchange of the first stage, we have

$$\begin{aligned} & SX^2(i, (t_{n-1}, t_{n-2})) \\ &= (K \times SX(i, (t_{n-1})) + t_{n-2}) \bmod N \\ &= (K((Ki + t_{n-1}) \bmod N) + t_{n-2}) \bmod N \\ &= ((K^2 i + K t_{n-1}) \bmod (NK) + t_{n-2}) \bmod N. \end{aligned}$$

Let $(K^2 i + K t_{n-1}) \bmod (NK) = m$, and then, $K^2 i + K t_{n-1} = CNK + m$, where C is a constant ≥ 0. Since $(m + t_{n-2}) \bmod N = (CNK + m + t_{n-2}) \bmod N$,

$$\begin{aligned} & ((K^2 i + K t_{n-1}) \bmod (NK) + t_{n-2}) \bmod N \\ &= (m + t_{n-2}) \bmod N \\ &= (CNK + m + t_{n-2}) \bmod N \\ &= (K^2 i + K t_{n-1} + t_{n-2}) \bmod N. \end{aligned}$$

Therefore,

$$\begin{aligned} & SX^2(i, (t_{n-1}, t_{n-2})) \\ &= ((K^2 i + K t_{n-1}) \bmod (NK) + t_{n-2}) \bmod N \\ &= (K^2 i + K t_{n-1} + t_{n-2}) \bmod N. \end{aligned}$$

Using mathematical induction on the stage number, we can show that at the stage n, the output terminal of the stage reachable from input terminal i is given by

$$SX^n(i, T) = (K^n i + K^{n-1} t_{n-1} + \ldots + K t_1 + t_0) \bmod N,$$

where the sequence of routing tags $T = (t_{n-1}, \ldots, t_0)$. □

Theorem 1 *In a K-ary shuffle-exchange network whose size, N, is a multiple of K, a path can be set up from input terminal i to output terminal j of stage n if the routing tag T_n specified by*

$$T_n = CN + j - K^n i,$$

exists such that $0 \leq T_n < K^n$ for some $C \geq 0$.

Proof: From Lemma 3, we know that output terminal j at the stage n can be reached from terminal i with routing tag T_n, such that

$$j = SX^n(i, T) = (K^n i + T_n) \bmod N,$$

where $T_n = K^{n-1}t_{n-1} + \ldots + Kt_1 + t_0$.
From the above equation,

$$K^n i + T_n = CN + j \quad \text{for some } C \geq 0$$

Thus, routing tag T_n can be specified by

$$T_n = CN + j - K^n i,$$

such that $0 \leq T_n = K^{n-1}t_{n-1} + \ldots + Kt_1 + t_0 < K^n$.
Therefore, if such T_n exists, then there is a path between i and j. □

The theorem establishes the relation between a routing tag, an input terminal and an output terminal of *any* stage. As long as the equation and the conditions associated with it are satisfied, we can have several different routing tags for paths between i and j. For example, for given $n, i,$ and j, we can have several Cs, if the resulting T_ns satisfy the condition, $0 \leq T_n < K^n$. These different routing tags lead a message to different paths. However, in other cases, we may not find T_n satisfying the condition for any possible Cs. This implies that there is no path between i and j for a given n. In such cases, we need to increase n to find T_n, that is, a message need to travel more hops than n to reach its destination. Note that n, the number of hops, is not fixed to $\lceil \log_K N \rceil$, and it can be another parameter.

Multiple Paths for Fault-tolerance

Multistage networks can be made to have multiple paths for fault-tolerance. The number of multiple paths can be increased by adding more stages to a multistage network. It is important to know the number of stages to guarantee a certain number of multiple paths for all pairs of source and destination. The following corollary of Theorem 1 shows the number of stages needed to guarantee M different paths for all source and destination pairs. Note that when $M = 1$, the corollary is specifying the number of stages for the multistage network to have *at least* one path between any source and destination.

Corollary 1 (The number of paths) *M different paths for any source and destination pair in a K-ary multistage shuffle-exchange network exist if the number of stages is larger than or equal to $\log_K(MN)$ where N, the number of nodes, is a multiple of K and $M \geq 1$.*

Proof: Since different routing tags set up different paths, we show that M different routing tags exist for any source and destination pair under the certain conditions.
From Theorem 1, we know that for source i and destination j, a routing tag T_n should satisfy the condition, $0 \leq T_n = CN + j - K^n i < K^n$. That is equivalent to

$$K^n i \leq CN + j < K^n i + K^n, \text{ for some } C \geq 0.$$

Let C_0 be the smallest C satisfying the condition $K^n i + K^n \leq C_0 N + j$, i.e., $(C_0 - 1) \times N + j < K^n i + K^n$.
If $K^n \geq N$, then $K^n i + N \leq K^n i + K^n \leq C_0 N + j$. By subtracting N from each side, we show that $K^n i \leq (C_0 - 1) \times N + j$. Therefore, we have a T_n satisfying $K^n i \leq T_n = (C_0 - 1) \times N + j < K^n i + K^n$. Since we can have such C_0 for each i and j pair, the condition, $K^n \geq N$, guarantees at least one T_n for any i and j pair. By using similar derivation, we can show that if $K^n \geq MN$, then

$$K^n i \leq C_p N + j < K^n i + NM \leq K^n i + K^n$$

such that $C_p = C_0 - 1, \ldots, C_0 - M$.
Hence, the condition, $K^n \geq MN$, guarantees M different T_n for any i and j pair. Therefore, if the number of stages, $n = \log_K(K^n) \geq \log_K(MN)$, then there are M different paths between any source and destination. □

2.3 Single Stage Shuffle-Exchange Networks

Multistage networks require a certain number of stages to provide the connectivity for all sources and destinations. On the other hand, single stage networks have all nodes connected even with only one stage, because unlimited number of traversals of the stage is possible. A smaller number of traversals is sufficient for some pairs of source and destination, although not for all pairs. For a given source and destination, it is possible to have multiple paths, and even multiple shortest paths. In this section, we will derive an algorithm to find a routing tag for a shortest path in a single stage network.

From Theorem 1, we know that if $K^n i \leq CN + j < K^n(i+1)$ is satisfied, then a path between i and j exist with the routing tag $T_n = CN + j - K^n i$. Since n means the number of hops to reach destination, T_n that satisfies the condition with the smallest n is the routing tag for a shortest path. By using an iterative method, we can find the smallest n for the number of hops in the shortest path between i and j.

Algorithm 1 *Shortest Path Routing Tag Generation*

Input :
  i : source node, j : destination node.
  K : shuffle degree, N : total number of nodes.
Output :
  T_n : a shortest path routing tag.

$\mathcal{A} \leftarrow Ki$, $\mathcal{B} \leftarrow K \times (i+1)$, $\mathcal{J} \leftarrow j$
while (1) {
  if $\mathcal{A} \leq \mathcal{J}$ *then*
    if $\mathcal{J} < \mathcal{B}$ *then*
      $T_n \leftarrow \mathcal{J} - \mathcal{A}$,
      return T_n.
    else
      $\mathcal{A} \leftarrow K\mathcal{A}$, $\mathcal{B} \leftarrow K\mathcal{B}$.
    endif
  else
    $\mathcal{J} \leftarrow \mathcal{J} + N$.
  endif
}

We will show that the alogrithm finds a shortest path routing tag in the following Lemma.

Lemma 4 (Shortest Path Routing) *Algorithm 1 finds a routing tag for a shortest path between node i and j in a single stage shuffle-exchange network.*

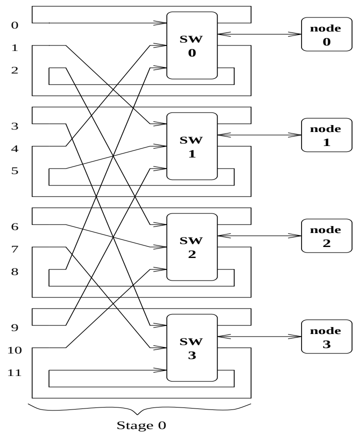

Figure 3: Direct Connected Single Stage Shuffle-Exchange Network

Proof: In order to find a shortest path routing tag, we should find C and the smallest n satisfying $K^n i \leq CN+j < K^n(i+1)$. When the condition is satisfied, the shortest path routing tag is $T_n = CN + j - K^n i$. For given C and n, n should be increased if $K^n(i+1) \leq CN+j$, and C should be increased if $CN + j < K^n i$. In each iteration of the algorithm, these conditions are examined at line 3 and 4, and only either n or C parameter is increased at line 8 and 11 one at a time. Therefore, n is increased only when there is no smaller n satisfying $K^n i \leq CN+j < K^n(i+1)$ with the current C and all previous Cs, and as soon as n is found to satisfy the condition, the algorithm returns the T_n obtained with the n and C, which is a shortest path routing tag. □

Direct Connected Single Stage Shuffle-Exchange Network

Single shuffle-exchange stage networks can take a different structure described in [15] as a directly connected single stage shuffle-exchange network(Direct SSX). So far, we assumed that in a single stage network a node injecting a message is connected to an input terminal of a stage. In this case, the message takes the shuffle permutation of links before the exchange function is performed at a switch. In a direct SSX, a node is connected to a switch itself instead of its input terminal, and each node is directly connected to other neighboring nodes via the switch as shown in Figure 3. In this structure, when a message is initially routed from a node, the exchange function of the switch is performed before the shuffle function is executed through the shuffle connection between switches. Logically, it can be represented by a *generalized de Bruijn Digraphs* defined in [5].

Since the application order of the shuffle and exchange function is reversed, the routing tag generation for the direct SSX networks may require a different algorithm. However, as it will be proven in the next corollary, the relation between routing tags and other parameters remain the same. The only differences are that N does not need to be a multiple of K because it represents the number of nodes instead of the number of total links as in the previous theorem. Also the source and destination are represented in node number instead of the input and output terminal numbers. Therefore, Algoritm 1 can be applied to the shortest path routing of direct SSX networks without modification.

We first define the connection structure of a direct single stage network.

Definition 2 (Direct SSX connection) *Let N be the number of nodes in a direct SSX network, and the size of the switching element in a node is K. The connection between nodes i and j can be specified by*

$$j \leftarrow (Ki + t) \bmod N$$

where node i is connected to node j through its output port t.

From the definition, a message injected from node i can reach node j through the output port t of node i after a hop. Note that t is added to Ki before the modulo N function is applied.

Corollary 2 (Routing in Direct SSX) *In a direct connected single stage K-ary shuffle-exchange network whose size is N, a path can be set up from source node i to destination node j at the n–th hop if the routing tag T_n specified by*

$$T_n = CN + j - K^n i,$$

exists such that $0 \leq T_n < K^n$ for some $C \geq 0$.

Proof: The proof is similar to the one used for theorem 1. From the definition of a connection, node i can set up a path to j by a routing tag t_n such that

$$j = (Ki + t_n) \bmod N, \quad 0 \leq i, j < N.$$

By using the same derivation as in Lemma 3, we can show that given a routing tag of $(t_n, t_{n-1}, \ldots, t_0)$, the destination node j will be

$$j = (K^n i + K^{n-1} t_{n-1} + \ldots + Kt_1 + t_0) \bmod N.$$

Thus, the routing tag $T_n (= K^{n-1}t_{n-1} + \ldots + Kt_1 + t_0)$ can be specified by

$$T_n = CN + j - K^n i, \text{ for some } C \geq 0.$$

such that $0 \leq T_n = K^{n-1}t_{n-1} + \ldots + Kt_1 + t_0 < K^n$. Therefore, if such T_n exists, then there is a path between i and j. □

3 Performance Evaluation

Shortest path routing provides several advantages over multistage routing. First, it eliminates all cycles in a message path. Since there is no self-cycle, single stage networks become less susceptible to deadlock. Second, the routing scheme provides the shortest internode distance for a given source and destination. The reduction in internode distance can boost network performance because messages traverse fewer hops, and network contention may be decreased.

3.1 Average Internode Distance

In our evaluation, we will focus on the effects on network performance of shorter internode distance resulting from the shortest path routing scheme. We will examine the average internode distances of single stage networks for the *shortest path routing* scheme and the *multistage routing* scheme by using various switch and network sizes. The average internode distance is an expected number of hops a message has to traverse when destination and source nodes are selected randomly. Therefore, under random traffic, the average internode distance indicates network performance.

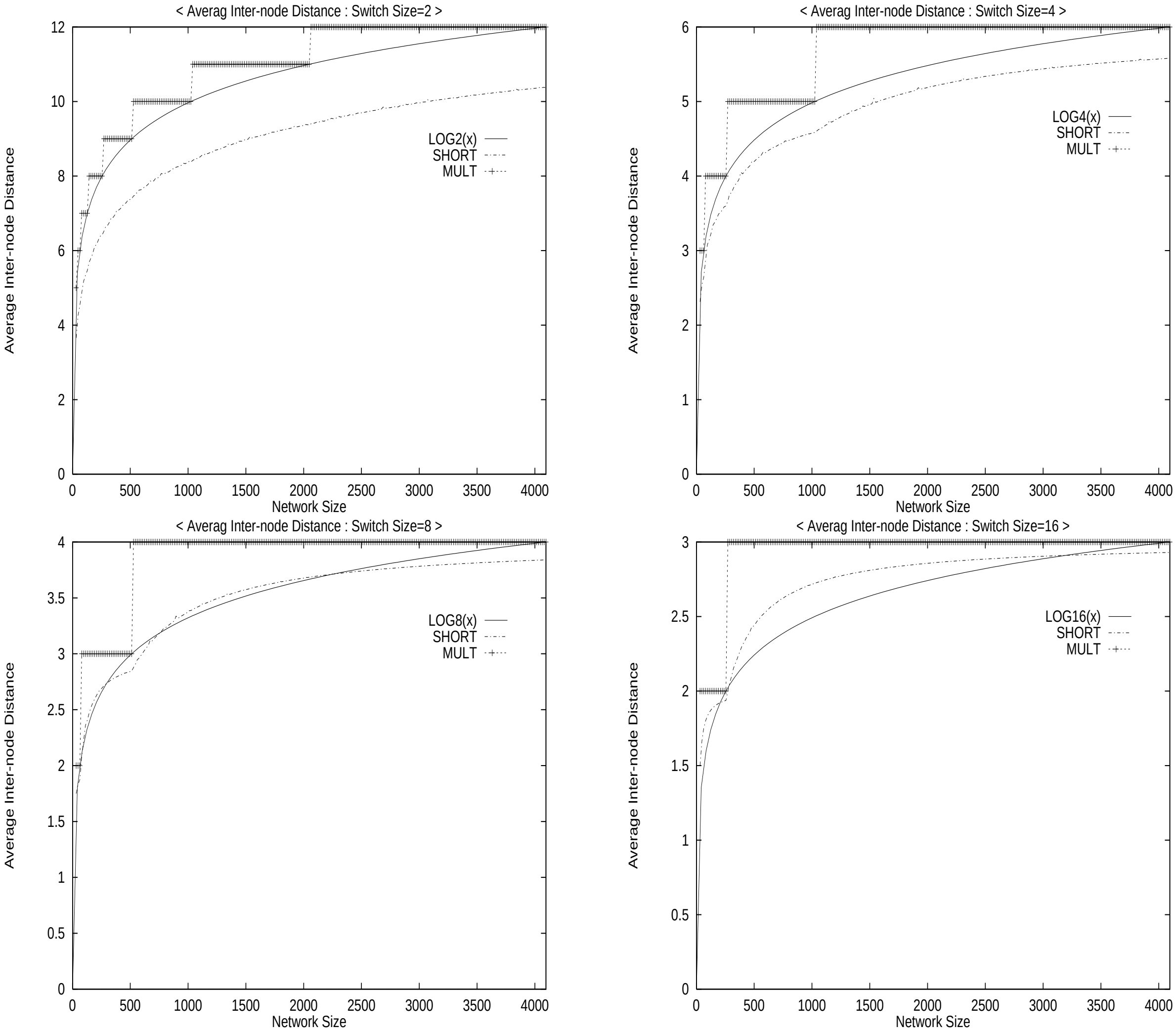

Figure 4: Average Internode Distance: SHORT=Shortest Path Routing, MULT=Multistage Routing, LOGN(X)=$log_N(X)$

The graphs in Figure 4 show the patterns of the average internode distance increase for the two different routing schemes and various switch sizes. The network size is increased from 32 to 4096 in increments of 16. In multistage routing, the average internode distance changes as a step function of the network size, such that the average internode distance jumps to the next natural number after the network size passes the power of the switch size. In shortest path routing, the increase in the average internode distance is closer to a logarithmic function of the network size. For some network and switch sizes, the average internode distance is less than the logarithmic value of the network size. Particularly, for the switch sizes 2 and 4, the average internode distance is less than the logarithmic value for all network sizes. Also note that when the switch size is smaller, the average internode distance increases more smoothly.

In Figure 5, we measure the ratio of the average internode distance in shortest path routing to that in multistage routing. The ratio indicates the relative performance of these routing schemes. The ratio is as low as around 66 % and as high as around 97 % for the network sizes from 32 to 4096. A lower ratio imply better performance of the shortest path routing. Drops in the ratios are found when the network sizes are near the next increment to the powers of the switch size. This is because multistage routing is most inefficient at these network sizes. When a switch size is smaller, there is a larger number of network sizes which have the drops in the ratios.

These results indicate that shortest path routing can provide shorter average internode distance and result in more gradual changes in network performance as the network size increases. The routing scheme may improve network performance the most when the network size is just over a power of the switch size. Also, a smaller switch size tends to result in bigger improvements for a wider range of network size with the shortest path routing scheme.

Although the average internode distance reveals certain properties of shortest path routing, we still need to know how much these properties affect actual system performance improvement. In order to measure such improvement, we employ trace-driven simulation of an entire shared memory system that includes processors, caches, memories and a single stage interconnection network. By using the system simulation instead of evaluating a network in isolation, we can accurately evaluate the effects of shortest path routing on memory system performance. In the next section, we will

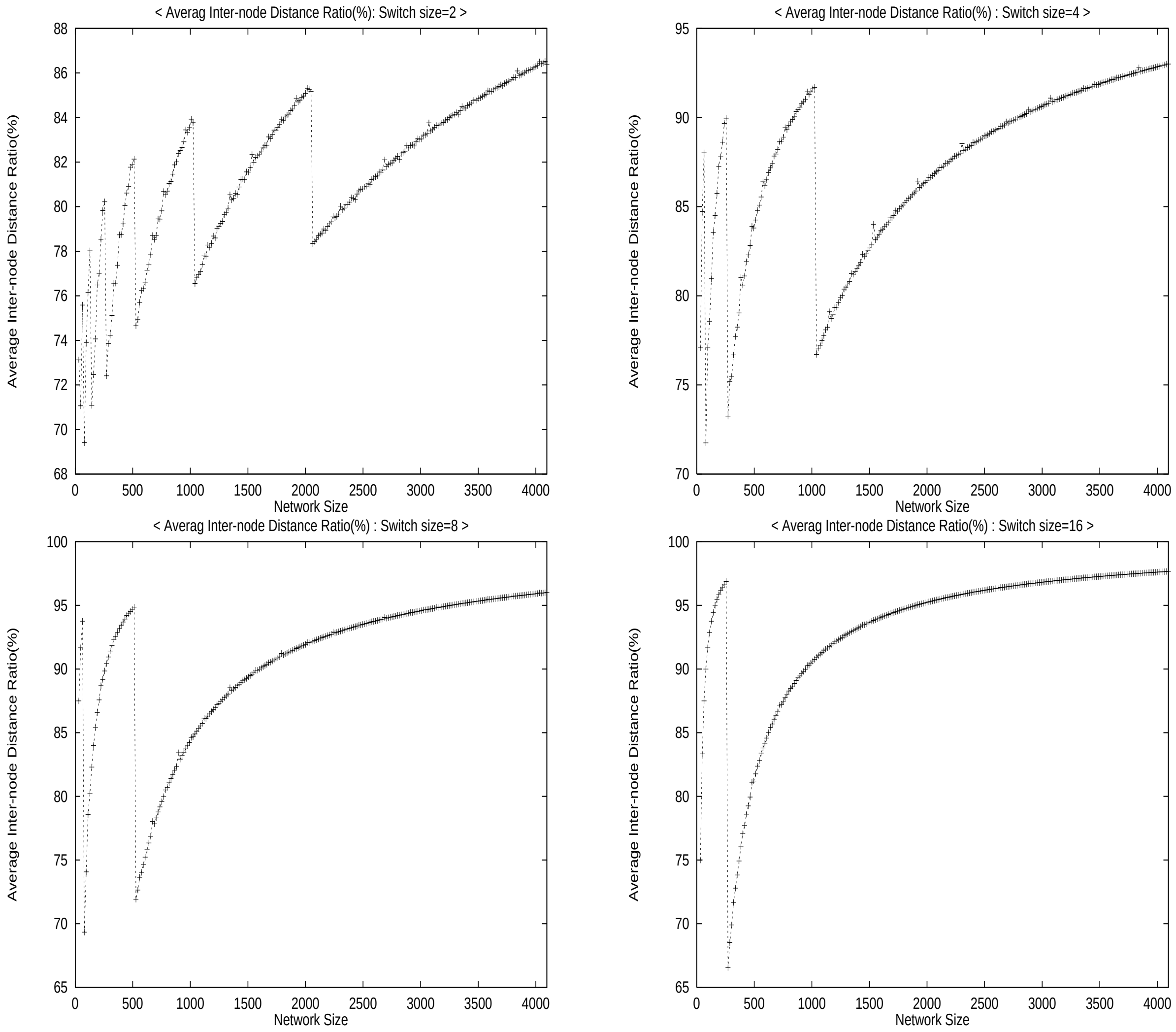

Figure 5: Average Internode Distance Ratio: Shortest Path Routing versus Multistage Routing

discuss our simulation methodology and analyze the results.

3.2 Evaluation Methodology

In this section, we describe our simulation methodology including models for each components, benchmarks and a tracing method.

Benchmarks and trace generation

For our evaluation, we selected the following benchmarks: ARC3D, FLO52, TRFD, MDG and MG3P. ARC3D is used to solve 3-D CFD problems. The next three codes are from the Perfect Club Benchmark suite. They are used to solve 2-D CFD problems, two-electron integral transformations and molecular dynamics problems, respectively. The last benchmark implements a 3-D multigrid algorithm. It is from NAS benchmark suite [1].

In some cases, instead of tracing a whole program, selected subroutines which account for a large fraction of execution time are traced. Initially, a parallel version of the programs that runs on CEDAR [7] multiprocessor was selected. All parallelism is expressed in terms of DOALL loops. These benchmarks were further modified to have better parallelism and cache performance through introduction of local variables. Also, loop interchange and unrolling were performed.

Table 1 shows the characteristics of each benchmark. The number of read requests is shown for 128 processor system. It varies with the number of caches in a system. The other metrics are the same regardless of system size.

The programs are instrumented by the MIPS tracing tool [4], compiled with a MIPS optimizing Fortran compiler, and executed on a MIPS M/120 system that has a MIPS R3000/3010 processor. The execution of the programs generates memory references and instruction execution time between such references, as well as information about parallel loop execution.

The trace is processed by a reordering tool that reorders loop iterations to emulate parallel loop execution, and then fed into our system simulator. The simulator consists of processors, caches, memories and a single stage network as shown in Figure 6.

Benchmark	Subroutines traced	Global Read Access (M)	Global Write Access (M)	Read Message Generated(M)	Write Message Generated(M)
FLO52	EFLUX	0.725123	0.263424	0.160802	0.263424
MDG	INTERF	4.550483	1.329371	1.098652	1.329371
TRFD	Whole Prog	7.628555	0.267484	0.353330	0.267484
ARC3D	STEPX	2.675342	0.561258	0.438169	0.561258
MG3P	Whole Prog	3.355304	0.192632	0.161779	0.192632

Table 1: Benchmark Characteristics

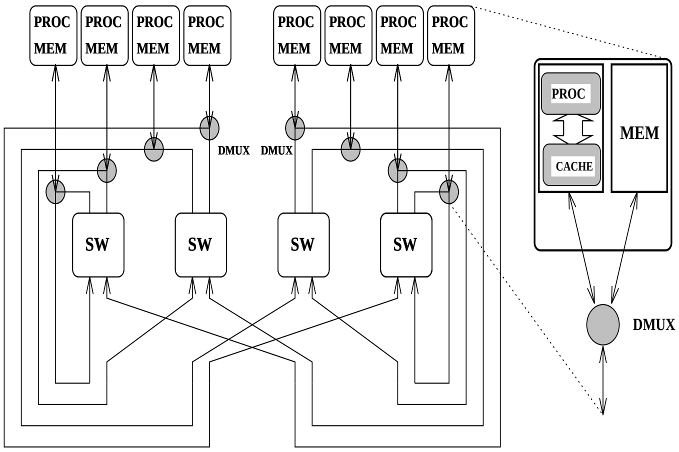

Figure 6: Single Stage Shuffle-Exchange Network System

Memory and caches

Each processor issues memory references according to the trace. The processor directly accesses a cache, and needs to send a message through a network to access main memory. The memory system models a shared memory system that follows the weak consistency model [6].

We use a direct mapped, write-through, no-write-allocate cache. The cache size is 8 Kbytes, the line size is 4 words, and word size is 8 bytes. The cache takes one cycle to service a read from a processor. A cache invalidation method [18] is employed to keep the caches coherent. A 64-entry write buffer is used to minimize processor stalls on writes. A write operation takes one cycle to put the write request into the write buffer. When a write buffer is full, a processor stalls on subsequent reads or writes. Only cache misses and writes generate request messages to a network. At most one outstanding read and one write request are allowed in the network simultaneously per cache and write buffer.

A read request message (8 bytes) consists of a message header only; a write request message (16 bytes) has a message header and one data word. A write reply message (8 bytes) from a memory has only a message header, and a read reply message (40 bytes) has a message header and a cache line that contains 4 words, and the requested word is returned first.

When synchronization is performed, all pending read and write requests are *completed* first. This includes issuing all write requests in a write buffer to a network and waiting for acknowledgment messages for all outstanding memory requests. Only then, synchronization proceeds.

In addition to caches, the memory system has interleaved and pipelined high-speed memory modules. These memory modules are interleaved by a cache line size. The access latency of a memory module is 4 cycles. The memory does not accept a new request message until it finishes receiving a request message from a network or sending its reply message back to the network. Thus, the memory cycle time is proportional to the length of a request message or a reply message whichever is longer.

We assume all local variables and instructions are stored in a *local* part of memories and have a perfect hit rate. Only memory references to shared variables are accessed through the network. This is based on our observation that the local accesses have 99% hit rates in these benchmarks.

Single stage shuffle-exchange network

The single stage shuffle-exchange network consists of only one shuffle-exchange stage as shown Fig 6. A processor and a memory module is attached to a link of the switch through a multiplexor and demultiplexor(DMUX). This construction is a relatively traditional approach and many papers on single stage shuffle-exchange networks are based on it [3, 12].

Non-blocking switches [8, 11, 16] with unlimited queue size are used. The switch supports the virtual-cut-through flow control scheme [9]. The channel width is 1 byte. A message is divided into a flits with size equal to the channel width, and then sent to the network. After arriving at a destination node, the message is assembled from the flits for further use. The minimum switch latency is 3 cycles.

The DMUX has also unlimited queue and minimum latency one cycle. Wire delay between the switches is one cycle as well.

3.3 Results and Analysis

In our experiments, we use the system size 64, 128, and 256 with switch sizes 4, 8, and 16. The average internode distances of single stage networks for shortest path routing and multistage routing are shown in Table 2. The column **SHORT** and **MULT** show the average internode distance of shortest path routing and that of multistage routing, respectively. The column **S/M** lists the ratio of the average internode distance of shortest path routing to that of multistage routing. The highest ratio, 96.88% occurs at the system size 256 with the switch size 16, and the lowest ratio, 80.21% occurs at the system size 128 with the switch size 8.

Table 3 shows read miss latencies of the single stage network systems, which are measured through the system simulations. The columns **SHORT, MULT** and **S/M** shows the read miss latency for shortest path routing, multistage routing and their ratios, respectively. We chose read miss latency as a major performance metric in the comparison of the routing schemes. Although the metric is affected by both network and memory performance, we can evaluate the effect of the routing schemes on the memory system performance.

As we see in the Table 3, for all benchmarks, shortest path routing provides better memory system performance, a shorter read miss latency. The improvements varies depending on different benchmarks. However, we can find a close relation between the **S/M** ratios in the Table 2 and Table 3. The ratios can not be the same because the read miss latency ratios in Table 3 contain the delay caused by memory access as well as network itself, whereas the **S/M** ratios in Table 2 reflect only network effects. The memory access delay is relatively constant and independent of the routing schemes. A large memory access delay reduces the relative difference in read miss latency for both routing schemes and results in a higher **S/M** ratio. Therefore, the **S/M** ratios in Table 3 tends to be larger than those in Table 2 because of the constant delay attributed to memory access delay. For example, the **S/M** ratio of MDG is affected the most by the memory access delays. This is the reason why the **S/M** ratios of MDG and the **S/M** ratio in Table 2 shows bigger differences than in other benchmarks. MDG has a hot spot memory access pattern so that memory access time is a dominant delay component in read miss latency. The difference between **S/M** ratios of the other benchmarks and those of the average internode distance is very close.

From these comparison studies, we expect the network performance improvement caused by shortest path routing to be reflected in the overall memory system performance. Therefore, for certain system and switch sizes, we can expect more than 30 % performance improvement.

4 Conclusions

We provided theoretical background and an routing tag generation algorithm for shortest path routing in single stage networks. The shortest path routing scheme can be applied to any network whose size is a multiple of a crossbar switch size. We extend the results to show a condition necessary to have a given number of multiple paths for all source and destination pairs for multistage networks. The routing scheme and routing tag generation algorithm can be used for direct connected single stage network architectures, *generalized de Bruijn Digraphs*, without modification.

The shortest path routing can improve single stage network performance in several aspects. By eliminating self-cycles in a message path, it reduces deadlock situations and also simplifies conflict resolution hardware. Most importantly, it results in a shorter average internode distance which directly improves network performance.

We compared average internode distance in single stage networks for shortest path routing and multistage routing, and studied the effects of shortest path routing on average internode distance. Average internode distance is increased more gradually with shortest path routing and results in shorter distance than with multistage routing. The most improvement in average internode distance with shortest path routing over multistage routing occurs when the network size is near the next to a power of a switch size. A smaller switch tends to have bigger improvements for a wider range of network size.

By using simulation studies, we evaluated how the shorter internode distance affects system performance. Our evaluation revealed a close relationship between the average internode distance improvements due to shortest path routing and the overall memory system performance improvement for our benchmarks. From these experiments, we can predict that shortest path routing can improve system performance up to more than 30 % depending on system and switch sizes.

References

[1] D. Bailey, H. Simon, J. Barton, and T.Lasinski. The NAS parallel benchmarks. Technical Report RNR-91-02, NASA Ames Research Center, 1991.

[2] Laxmi N. Bhuyan and Dharma P. Agrawal. Design and performance of generalized interconnection networks. *IEEE Transactions on Computers*, C-32(12):1081–1090, December 1983.

[3] Pin-Yee Chen, Pen-Chung Yew, and Duncan Lawrie. Performance of packet switching in buffered single-stage shuffle-exchange networks. In *International Conference on Distributed Computing Systems*, pages 622–627, 1982.

[4] Yung-Chin Chen. *Cache Design and Performance in a Large-Scale Shared-Memory Multiprocessor System.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[5] D. Z. Du and F. K. Hwang. Generalized de bruijn diagraphs. *Networks*, 18:27–38, 1988.

[6] M. Dubois, Scheurich C, and F. Briggs. Memory access buffering in multiprocessors. In *International Symposium on Computer Architecture*, pages 434–442, 1986.

[7] D. Gajski, D. Kuck, D. Lawrie, and A. Sameh. Cedar - a large scale multiprocessor. In *International Conference on Parallel Processing*, pages 524–529, 1983.

[8] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, COM-35(12):1347–1356, December 1987.

[9] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[10] Clyde P. Kruskal and Marc Snir. A unified theory of interconnection network structure. *Theoretical Computer Science*, 48(1):75–94, August 1986.

[11] Manoj Kumar and J. R. Jump. Performance enhancement in buffered delta networks using crossbar switches and multiple links. *Journal of Parallel and Distributed Computing*, (1):81–103, 1984.

[12] D. H. Lawrie and D. A. Padua. Analysis of message switching with shuffle-exchanges in multiprocessors. In *The Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing*, pages 116–123, 1980.

[13] Duncan H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):173–182, December 1975.

[14] Krishnan Padmanabhan. Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):385–397, October 1991.

Switch Size	System Size								
	64			128			256		
	SHORT	MULT	S/M %	SHORT	MULT	S/M %	SHORT	MULT	S/M %
04	2.64	3	88.02	3.24	4	80.96	3.6	4	89.97
08	1.88	2	93.75	2.41	3	80.21	2.7	3	89.84
16	1.75	2	87.50	1.88	2	93.75	1.94	2	96.88

Table 2: Average Internode Distance: SHORT=Shortest Path Routing, MULT=Multistage Routing, S/M = Ratio of Shortest Path Routing to Multistage Routing

Switch Size	System Size								
	64			128			256		
	SHORT	MULT	S/M (%)	SHORT	MULT	S/M (%)	SHORT	MULT	S/M (%)
ARC3D									
04	115.7	134.8	85.81	154.4	190.7	80.94	179.4	196.3	91.39
08	79.97	86.38	92.59	112.4	137.9	81.52	131	144	91
16	76.14	86.25	88.28	84.55	89.51	94.46	95.89	98.04	97.81
EFLUX									
4	217.4	250.4	86.81	276.9	330.2	83.85	311.7	338	92.22
8	149.4	160.4	93.16	205.8	246.7	83.43	229.7	246.9	93.05
16	141.1	159.5	88.46	149.9	158.8	94.38	160.3	164.3	97.56
MDG									
04	218.5	238.9	91.45	292.8	332.4	88.09	329.9	350.6	94.11
08	177.8	185.5	95.89	257.6	279.7	92.11	271.3	279.7	97.01
16	172.2	185.2	92.98	224.1	230.3	97.31	246.7	250.2	98.58
MG3P									
04	129.6	150.4	86.17	183.2	226.3	80.94	219.2	240.6	91.13
08	88.79	96.41	92.1	134.6	163.2	82.47	164.3	178.6	92
16	83.26	95.27	87.39	99.05	105.8	93.66	117.3	120.9	97.01
TRFD									
04	120.1	134.2	89.48	145.1	170.7	84.97	162.1	177.3	91.44
08	90.33	96.15	93.95	113.3	132.7	85.34	129.5	140	92.45
16	86.36	95.68	90.26	90.84	95.99	94.63	99.89	102.2	97.7

Table 3: Read Miss Latency: SHORT=Shortest Path Routing, MULT=Multistage Routing, S/M = Ratio of Shortest Path Routing to Multistage Routing

[15] Krishnan Padmanabhan. On the tradeoff between node degree and communication channel width in shuffle-exchange networks. In *Proceedings, IEEE Symposium on Parallel and Distributed Processing*, pages 120–127. IEEE, December 1991.

[16] Ora E. Percus and Susan R. Dickey. Performance analysis of clocked-regulated queues with output multiplexing in three different 2x2 crossbar switch architectures. *Journal of Parallel and Distributed Computing*, (16):27–40, 1992.

[17] Xiao-Nan Tan and Kenneth C.Sevcik. Reduced distance routing in single-stage shuffle-exchange interconnection networks. Technical report, University of Toronto, September 1986.

[18] Alexander Veidenbaum. A compiler-assisted cache coherence solution for multiprocessors. In *International Conference on Parallel Processing*, pages 1029–1036, 1986.

[19] Pen-Chen Yew. *On The Design of Interconnection Networks For Parallel and Multiprocessor Systems.* PhD thesis, University of Illinois at Urbana-Champaign, 1981.

[20] Pen-Chung Yew, David A. Padua, and Duncan H. Lawrie. Stochastic properties of a multiple-layer single-stage shuffle-exchange network in a message switching environment. *Journal of Digital Systems*, VI(4):387–410, 1980.