Optimal All-to-All Personalized Exchange in Self-Routable Multistage Networks

Yuanyuan Yang, *Senior Member*, *IEEE Computer Society*, and
Jianchao Wang, *Member*, *IEEE Computer Society*

Abstract—All-to-all personalized exchange is one of the most dense collective communication patterns and occurs in many important applications in parallel computing. Previous all-to-all personalized exchange algorithms were mainly developed for hypercube and mesh/torus networks. Although the algorithms for a hypercube may achieve optimal time complexity, the network suffers from unbounded node degrees and thus has poor scalability in terms of I/O port limitation in a processor. On the other hand, a mesh/torus has a constant node degree and better scalability in this aspect, but the all-to-all personalized exchange algorithms have higher time complexity. In this paper, we propose an alternative approach to efficient all-to-all personalized exchange by considering another important type of networks, multistage networks, for parallel computing systems. We present a new all-to-all personalized exchange algorithm for a class of unique-path, self-routable multistage networks. We first develop a generic method for decomposing all-to-all personalized exchange patterns into some permutations which are realizable in these networks, and then present a new all-to-all personalized exchange algorithm based on this method. The newly proposed algorithm has $O(n)$ time complexity for an $n \times n$ network, which is optimal for all-to-all personalized exchange. By taking advantage of fast switch setting of self-routable switches and the property of a single input/output port per processor in a multistage network, we believe that a multistage network could be a better choice for implementing all-to-all personalized exchange due to its shorter communication latency and better scalability.

Index Terms—Parallel computing, interprocessor communication, collective communication, all-to-all communication, all-to-all personalized exchange, routing, multistage interconnection networks, Latin Square, permutation.

1 Introduction

Collective communication [1], [2] involves global data movement and global control among a group of processors in a parallel/distributed computing system. Many scientific applications exhibit the need of such communication patterns. Efficient support for collective communication can significantly reduce the communication latency and simplify the programming of parallel computers. Collective communication has received much attention in parallel processing community in recent years; see, for example, [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15].

Among all collective communication operations, *all-to-all communication* is one of the most dense operations. In all-to-all communication, every processor in a processor group sends a message to all other processors in the group. Depending on the nature of the message to be sent, all-to-all communication can be further classified as *all-to-all broadcast* and *all-to-all personalized exchange*. In all-to-all broadcast, every node sends the same message to all other nodes, and in all-to-all personalized exchange, every node sends a distinct message to every other node. Clearly, all-to-all broadcast can be viewed as a special case of all-to-all personalized exchange. Both all-to-all broadcast and all-to-all personalized exchange are used in many parallel applications. For example, all-to-all broadcast is needed in matrix multiplication, LU-factorization and Householder transformations, and all-to-all personalized exchange is needed in matrix transposition and Fast Fourier Transform (FFT) [16], [17]. In this paper, we will be mainly interested in efficient algorithms for all-to-all personalized exchange.

There has been much work for all-to-all personalized exchange in various networks; see, for example [8], [9], [10], [11], [12], [13], [14], [15], [16]. Johnsson and Ho [8] proposed optimal all-to-all personalized exchange algorithms on an n-node hypercube with $O(n \log n)$ and $O(n)$ time complexity for one-port model and all-port model, respectively. However, a drawback of high-dimensional networks such as hypercubes is their poor scalability due to the unbounded node degree, which corresponds to the number of I/O ports needed for interprocessor communication in a processor. On the other hand, meshes and tori have a simple, regular topology and a bounded node degree, and become more and more popular for interconnecting processors in parallel/distributed computing systems due to their better scalability in terms of I/O port limitation in a processor. Typical all-to-all personalized exchange algorithms on a two dimensional mesh and torus [11], [12], [13], [14], [15], [16] have time complexity $O(n^{\frac{3}{2}})$, where n is the number of nodes in the network. In general, for a k dimensional mesh and torus, the algorithms have time complexity $O(n^{\frac{k+1}{k}})$ (see, for example, Suh and Shin [15]). Apparently, using a mesh or a torus for all-to-all personalized exchange suffers a

- *Y. Yang is with the Deptpartment of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794. E-mail: yang@ece.sunysb.edu.*
- *J. Wang is with GTE Laboratories, 40 Sylvan Road, Waltham, MA 02454. E-mail: jwang@gte.com.*

Manuscript revised 17 Sep. 1999; accepted 23 Oct. 1999.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 107421.

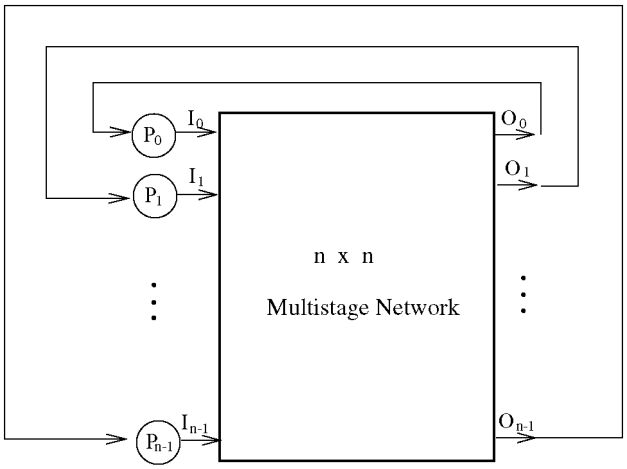

Fig. 1. Processors communicating through a multistage network.

longer communication delay than hypercubes due to the limitations of the network topologies themselves.

In this paper, we consider another important type of interconnecting scheme of parallel computing systems, multistage interconnection networks (MINs). In particular, we focus on a class of unique-path, self-routable MINs (for example, baseline, omega, indirect binary n-cube (or banyan) network, and the reverse networks of these networks, etc.). Due to their easy routing capability and uniform communication latency between any network inputs and outputs, this type of MIN has been used in several parallel computers, such as NEC Cenju-3 and IBM SP. There has been some work in the literature on collective communication in self-routable MINs. For example, [4], [5], [6] proposed different approaches for multicast/broadcast in wormhole routed MINs. However, to our knowledge, no one has studied all-to-all personalized exchange in this type of MINs, which is the main focus of this paper. As can be seen later, the newly proposed all-to-all scheme is suitable for both packet switched and wormhole routed MINs.

Given n processors $P_0, P_1, \ldots, P_{n-1}$, an $n \times n$ multistage network can be used for interprocessor communication as depicted in Fig. 1. Each processor can send messages through an input of the network, and receive messages from an output of the network. Generally, the multistage network used can be chosen from a variety of networks, such as crossbar, Clos [18], Benes [19], baseline, omega [20], indirect binary n-cube [21], etc. Although a crossbar and a Clos network both can realize all possible permutations between the network inputs and outputs and have a constant communication latency, the network costs of an $n \times n$ crossbar and a three-stage Clos network are $O(n^2)$ and $O(n^{\frac{3}{2}})$, respectively, which are generally considered too high for a large system. A Benes network, which has a network cost $O(n \log n)$, also can realize all permutations, but not all permutations can be easily routed [22] through the network and some rearrangements of existing connections may be needed. In addition, the Benes network can be viewed as a concatenation of a baseline network and a reverse baseline network with the center stages overlapped, and thus has the network cost as well as the communication latency almost twice of those of a baseline network. Baseline, omega, indirect binary n-cube, etc. are a class of self-routable networks with a unique path between each input/output pair in the network. Although this type of network can realize only a proper subset of permutations, a full permutation capability may not be necessary for all-to-all personalized exchange. In this paper, we develop a generic method for decomposing all-to-all personalized exchange patterns into some permutations which are realizable in these networks, and present a new all-to-all personalized exchange algorithm based on this method. The newly proposed algorithm has $O(n)$ time complexity for an $n \times n$ network, which is optimal for all-to-all personalized exchange. By taking advantage of fast switch setting of self-routable switches and the property of single input/output port per processor in a multistage network, we believe that a multistage network could be better a choice for implementing all-to-all personalized exchange due to its shorter communication latency and better scalability.

The remainder of this paper is organized as follows. Section 2 reviews the definitions of the class of multistage networks under consideration and discusses some relevant properties of permutations. Section 3 presents a new all-to-all personalized exchange algorithm for a multistage network based on a Latin Square, whose each row corresponds to an admissible permutation of the network. Section 4 describes two methods of constructing the Latin Squares closely related to the class of multistage networks and used in the new algorithm. Section 5 discusses how to generate admissible permutations which can form the Latin Square in a generic way for all-to-all personalized exchange in the class of multistage networks. Section 6 summarizes the time complexity of the algorithm, and compares this new algorithm with the existing algorithms for other network topologies. Section 7 concludes the paper. Finally, the appendix gives some detailed proofs of theorems.

2 Network Structures and Permutations

Multistage interconnection networks such as baseline, omega, and indirect binary n-cube networks, have been proposed and widely used in parallel processing systems [20]. A typical network structure for this class of networks is that each network has $n(=2^m)$ inputs and outputs and $\log n(=m)$ stages, with each stage consisting of $\frac{n}{2}$ 2×2 switches and any two adjacent stages connected by n interstage links. Fig. 2a, Fig. 2b and Fig. 2c illustrate an 8×8 baseline network, omega network and indirect binary n-cube network, respectively.

A permutation is a one-to-one mapping between the network inputs and outputs. For an $n \times n$ network, suppose there is a one-to-one mapping ρ which maps input i to output a_i (i.e. $\rho(i) = a_i$), where $a_i \in \{0, 1, \ldots, n-1\}$ for $0 \le i \le n-1$, and $a_i \neq a_j$ for $i \neq j$. Let

$$\rho = \begin{pmatrix} 0 & 1 & \ldots & n-1 \\ a_0 & a_1 & \ldots & a_{n-1} \end{pmatrix}$$

denote this permutation. In particular, when $\rho(i) = i$ for $0 \le i \le n-1$, we refer to this permutation as an *identity permutation* and denote it as I.

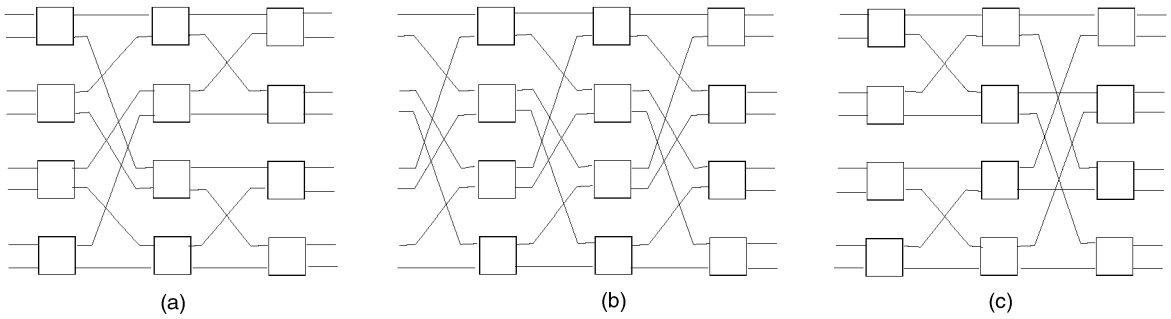

Fig. 2. Three typical multistage networks.

We now briefly review some properties and notations of permutations which will be used in this paper. Given two permutations ρ_1 and ρ_2, a composition $\rho_1\rho_2$ of the two permutations is also a permutation, which maps i to $\rho_1(\rho_2(i))$. Clearly, $\rho I = I\rho = \rho$, but in general $\rho_1\rho_2 \neq \rho_2\rho_1$. However, the associative law does apply here. That is, $\rho_1(\rho_2\rho_3) = (\rho_1\rho_2)\rho_3$. Let ρ^i denote the composition of i permutations ρs. Also, if $\rho_1\rho_2 = I$, we call ρ_1 the inverse of ρ_2 and vice versa, and denote them as $\rho_1 = \rho_2^{-1}$ and $\rho_2 = \rho_1^{-1}$. A permutation can also be expressed as a cycle or a composition of several cycles. For example, in a 4×4 mapping, a cycle $(0, 3, 2)$ represents a permutation in which $0, 3,$ and 2 are mapped to $3, 2,$ and 0, respectively, while 1 is kept unchanged. In addition, for representational convenience, we use the following notation to represent a mapping $\rho(a) = b$

$$a \xrightarrow{\rho} b.$$

In the context of a multistage network, each stage in the network can be viewed as a shorter $n \times n$ network, and so does each set of interstage links. Let σ_i $(0 \leq i \leq m-1)$ denote the permutation represented by stage i, and π_i $(0 \leq i \leq m-2)$ denote the permutation represented by the set of interstage links between stage i and stage $i+1$. We refer to the permutation σ_i as *stage permutation*, the permutation π_i as *interstage permutation*, and the permutation realized by the entire multistage network as *admissible permutation* of the network. Clearly, an admissible permutation can be expressed by a composition of stage permutations and interstage permutations. For example, the admissible permutation of a baseline network can be expressed as

$$\sigma_{m-1}\pi_{m-2}\sigma_{m-2}\ldots\pi_0\sigma_0. \tag{1}$$

In general, interstage permutations π_i's are fixed by the network topology. For a baseline network, suppose the binary representation of a number $a \in \{0, 1, \ldots, n-1\}$ is $p_{m-1}p_{m-2}\ldots p_1p_0$. Then the interstage permutation π_i of baseline network represents the following mapping:

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi_i} p_{m-1}p_{m-2}\ldots p_{m-i}p_0p_{m-i-1}\ldots p_2p_1 \tag{2}$$

This mapping corresponds to a 1-bit circular-right-shift among the $m-i$ least significant bits while keeping the i most significant bits unchanged.

However, stage permutation σ_is are not fixed since each switch can be set to either parallel or cross. Thus σ_i can be a composition of any subset of cycles $\{(0,1),(2,3),\ldots,(n-1,n)\}$, which implies that there are a total of $2^{\frac{n}{2}}$ possible choices for each σ_i. It follows that by (1) the number of all admissible permutations of a baseline network is $\left(2^{\frac{n}{2}}\right)^{\log n} = n^{\frac{n}{2}}$.

To avoid confusions in the rest of the paper, π_is are used to represent the interstage permutations only for baseline network.

For an $n \times n$ omega network, each of the $\log n$ interstage permutations is a shuffle function which is exactly π_0^{-1}, where π_0 is defined in (2). In fact, π_0^{-1} is a 1-bit circular-left-shift operation, that is,

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi_0^{-1}} p_{m-2}p_{m-3}\ldots p_1p_0p_{m-1}. \tag{3}$$

The overall admissible permutation of an omega network is $\sigma_{m-1}\pi_0^{-1}\sigma_{m-2}\pi_0^{-1}\ldots\sigma_1\pi_0^{-1}\sigma_0\pi_0^{-1}$.

For an indirect binary n-cube network, let τ_i denote the interstage permutation between stage i and stage $i+1$ for $0 \leq i \leq m-2$. τ_i represents the following mapping

$$\begin{aligned} &p_{m-1}p_{m-2}\ldots p_{i+2}p_{i+1}p_i\ldots \\ &p_1p_0 \xrightarrow{\tau_i} p_{m-1}p_{m-2}\ldots p_{i+2}p_0p_i\ldots p_1p_{i+1}, \end{aligned} \tag{4}$$

which is the function of swapping bit 1 for bit $i+2$. Similar to a baseline network, the overall admissible permutation of an indirect binary n-cube network is $\sigma_{m-1}\tau_{m-2}\sigma_{m-2}\ldots\tau_0\sigma_0$.

Now let's look at an example of a baseline network shown in Fig. 3. We have stage permutations $\sigma_0 = (2,3)$, $\sigma_1 = (0,1)(4,5)$, and $\sigma_2 = (0,1)(2,3)(4,5)(6,7)$, and interstage permutations (in both binary and decimal)

$$\begin{aligned} \pi_0 &= \begin{pmatrix} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 000 & 100 & 001 & 101 & 010 & 110 & 011 & 111 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 4 & 1 & 5 & 2 & 6 & 3 & 7 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \pi_1 &= \begin{pmatrix} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 000 & 010 & 001 & 011 & 100 & 110 & 101 & 111 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 1 & 3 & 4 & 6 & 5 & 7 \end{pmatrix} \end{aligned}$$

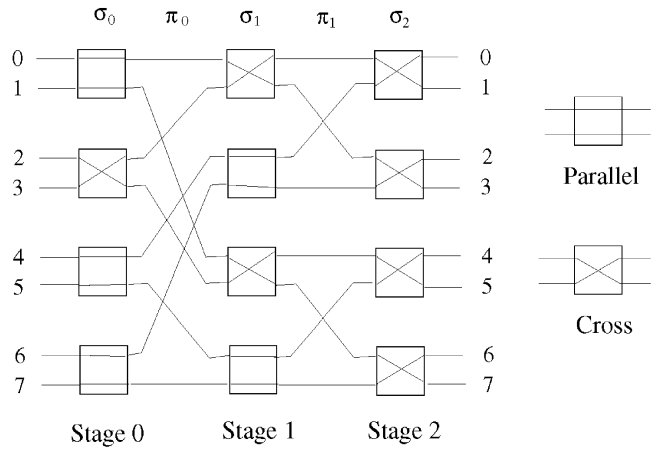

Fig. 3. A routing example in an 8×8 baseline network.

For input 0, we can obtain the following transformation

$$0 \xrightarrow{\sigma_0} 0 \xrightarrow{\pi_0} 0 \xrightarrow{\sigma_1} 1 \xrightarrow{\pi_1} 2 \xrightarrow{\sigma_2} 3,$$

that is,

$$0 \xrightarrow{\sigma_2\pi_1\sigma_1\pi_0\sigma_0} 3.$$

After computing the transformation for every input, we can obtain the overall permutation for the switch settings in the network

$$\sigma_2\pi_1\sigma_1\pi_0\sigma_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 5 & 1 & 0 & 4 & 2 & 6 \end{pmatrix}.$$

Finally, we review the self-routable property of this class of MINs. Self-routing is a fast routing scheme in which the routing decision at a switch depends only on the addresses of source and destination, which form the routing tags. We still take a baseline network as an example. Suppose the destination address is $d_{m-1}d_{m-2}\cdots d_1d_0$ in binary format. At stage i $(0 \le i \le m-1)$ the message is routed from an input of a switch to the upper or lower output of the switch depending on the routing tag d_{m-i-1} being 0 or 1. We can see that in Fig. 3, to route a message from input 5 to output 4 (100), the switch settings for three stages are "down," "up," and "up," respectively. Clearly, the time complexity of switch setting at each stage is a small constant. The self-routing algorithms for other networks in this class are slightly different but similar, which can be found in many books on interconnection networks such as [1].

3 Realizing All-to-All Personalized Exchange in Multistage Networks

As mentioned earlier, we intend to realize all-to-all personalized exchange in a multistage network of $\log n$ stages. In this section, we first discuss the lower bound on the communication time for all-to-all personalized exchange in such a network, and then propose an optimal algorithm for realizing all-to-all personalized exchange.

3.1 Lower Bound for All-to-All Personalized Exchange

We have the following lemma concerning the lower bound on the maximum communication delay of all-to-all personalized exchange in a multistage network.

Lemma 1. *The maximum communication delay of all-to-all personalized exchange in an $n \times n$ network of $\log n$ stages is at least $\Omega(n + \log n)$.*

Proof. The lemma holds because each processor must receive one message from all other $n-1$ processors, which takes $\Omega(n)$ time, and each message must go through $\log n$ stages from its source processor to its destination processor, which takes $\Omega(\log n)$ time. Since there is only one input port for each processor, it must take at least $\Omega(n + \log n)$ time to receive all the messages sent to it. □

3.2 All-to-All Personalized Exchange Algorithm Using a Latin Square

A *Latin Square* [23] is defined as an $n \times n$ matrix

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{bmatrix}$$

in which the entries $a_{i,j}$s are numbers in $\{0, 1, 2, \ldots, n-1\}$ and no two entries in a row (or a column) have the same value. We can also describe a Latin Square in a different way: for all i and j, $0 \le i, j \le n-1$, the entries of each row in the matrix, $a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}$, form a permutation

$$\begin{pmatrix} 0 & 1 & 2 & \cdots & n-1 \\ a_{i,0} & a_{i,1} & a_{i,2} & \cdots & a_{i,n-1} \end{pmatrix}$$

and the entries of each column in the matrix,

$$a_{0,j}, a_{1,j}, \ldots, a_{n-1,j},$$

also form a permutation

$$\begin{pmatrix} 0 & 1 & 2 & \cdots & n-1 \\ a_{0,j} & a_{1,j} & a_{2,j} & \cdots & a_{n-1,j} \end{pmatrix}.$$

In this paper, we say two Latin Squares are *equivalent* if one can be transformed into another by swapping rows of its matrix. This concept is useful when we consider different approaches to constructing a Latin Square.

In this section, we assume that for a self-routable multistage network under consideration, there exists a Latin Square such that any permutation formed by each row of the matrix is admissible to the network; that is, this permutation is self-routable from the inputs to the outputs of the network. Under this assumption, we can design an all-to-all personalized exchange algorithm (ATAPE) which is generic for the class of multistage networks, with details of self-routing being omitted. We also assume that every message has the same length so that the message transmission at each stage is synchronized. A higher-level description of the algorithm ATAPE is given in Table 1.

In algorithm ATAPE, processor j sends distinct messages to all destinations in the order of $a_{0,j}$, $a_{1,j}$, …, $a_{n-1,j}$,

TABLE 1
All-to-All Personalized Exchange Algorithm for a Class of Multistage Networks

Algorithm ATAPE
begin
Step 1. for each processor j $(0 \le j \le n-1)$ **do in parallel**
  1.1 for each $a_{i,j}$ $(0 \le i \le n-1)$ in the Latin square **do in sequential**
    prepare a personalized message from processor j to processor $a_{i,j}$;
    insert the message into the message queue j;
Step 2. for each processor j $(0 \le j \le n-1)$ **do in parallel**
  2.1 for each message with destination address $a_{i,j}$ $(0 \le i \le n-1)$ in the message queue j **do in sequential**
    send the message destined to $a_{i,j}$ through input j of the network;
end;

which corresponds to column j of the Latin Square. On the other hand, in time frame i, all n processors send their messages simultaneously to destinations $a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}$, which corresponds to row i of the Latin Square. Thus, in algorithm ATAPE, all-to-all personalized exchange is achieved by realizing n permutations which correspond to the n rows of the Latin Square, since under the assumption that each permutation represented by a row of the Latin Square is admissible to the network.

Note that the network under consideration is a self-routable network, where each switch is set automatically by the routing tag contained in the message passing that switch. Since we are considering admissible permutations, at any time two messages entering from the two inputs of a switch can pass the switch simultaneously without any conflicts. In addition, once the previous n messages leave the switches of the current stage, the next n messages can enter the switches of this stage. Thus, the sequential steps of 2.1 are actually performed in a pipelined fashion, which achieves a form of parallelism. Therefore, the time complexities of Step 1 and Step 2 are $O(n)$ and $O(n + \log n)$, respectively. The total time delay for the all-to-all personalized exchange algorithm is $O(n + \log n)$.

From the description of the above algorithm, we can see that proposed approach is suitable for both packet switched and wormhole switched MINs.

Now, the only problem remains unsolved is how to construct the special Latin Square the all-to-all personalized exchange algorithm for this class of multistage networks is based on, which is the main focus of the rest of the paper.

4 Two Methods for Constructing a Latin Square

Although in general there are many ways to construct a Latin Square, in this section we describe two methods of constructing the Latin Squares suitable to the class of multistage networks we consider in this paper. The first method will be used in the next section to prove that a special set of admissible permutations of a multistage network does form a Latin Square, and the second method will be used to efficiently generate such admissible permutations.

First, we introduce a set of basic permutations used for constructing a Latin Square. For an $n \times n$ mapping, where $n = 2^m$, we define m basic permutations ϕ_i $(1 \le i \le m)$ as follows. Let the binary representation of a number $a \in \{0, 1, \ldots, n-1\}$ be $p_{m-1}p_{m-2}\ldots p_1p_0$. Then

$$p_{m-1}p_{m-2}\ldots p_ip_{i-1}p_{i-2}\ldots p_1p_0 \xrightarrow{\phi_i} p_{m-1}p_{m-2}\ldots p_i\bar{p}_{i-1}p_{i-2}\ldots p_1p_0. \quad (5)$$

The permutation ϕ_i is actually the operation flipping bit i of a binary number. ϕ_i can also be expressed as a composition of $\frac{n}{2}$ 2-cycles. For example, the three basic permutations for $n = 8$ are

$$\begin{aligned} \phi_1 &= (0,1)(2,3)(4,5)(6,7) \\ \phi_2 &= (0,2)(1,3)(4,6)(5,7) \\ \phi_3 &= (0,4)(1,5)(2,6)(3,7), \end{aligned}$$

which are also shown in Fig. 4. Although we can obtain a similar expression for the composition of 2-cycles for a general n, we would rather give an intuitive description of the basic permutations. The mapping of n numbers $0, 1, 2, \ldots, n-1$ by the basic permutation

$$\phi_i \ (1 \le i \le m = \log n)$$

is performed as follows (illustrated in Fig. 5): First divide the entire segment containing all n numbers into 2^{m-i+1} subsegments with each subsegment containing 2^{i-1} consecutive numbers; then starting from subsegment 0, group two consecutive subsegments into a pair; finally, swap two subsegments in each pair. Take ϕ_2 in Fig. 4 as an example. We divide eight numbers into four subsegments: $||0,1||$, $||2,3||$, $||4,5||$, and $||6,7||$. Then swap the first pair of consecutive subsegments $||0,1||$ and $||2,3||$, and also swap the second pair of consecutive subsegments $||4,5||$ and $||6,7||$. Thus, ϕ_2 maps $0,1,2,3,4,5,6,7$ to $2,3,0,1,6,7,4,5$.

Before we use the basic permutations in constructing Latin Squares, we discuss some nice properties of them.

Lemma 2. *The set of basic permutations ϕ_i $(1 \le i \le m)$ defined in (5) has the properties that the composition of any two basic permutations is exchangeable, and the composition of two identical basic permutations equals the identity permutation. That is,*

$$\phi_i\phi_j = \phi_j\phi_i, \ for\ 1 \le i, j \le m \quad (6)$$

and

$$\phi_i\phi_i = I, \ for\ 1 \le i \le m, \quad (7)$$

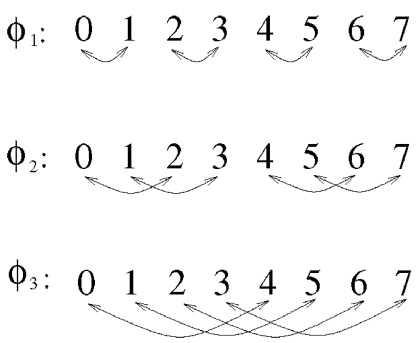

Fig. 4. Basic permutations for an 8×8 mapping. Each arc represents a mapping between two numbers.

Proof. The exchangeability (6) can be easily seen from the definition of the basic permutations. In fact, any binary number $p_{m-1}p_{m-2}\cdots p_{i-1}\cdots p_{j-1}\cdots p_1p_0$ can be mapped to $p_{m-1}p_{m-2}\cdots \bar{p}_{i-1}\cdots \bar{p}_{j-1}\cdots p_1p_0$ by either permutation $\phi_i\phi_j$ or permutation $\phi_j\phi_i$. Similarly, we can see (7) holds. This is because applying the composition of two ϕ_is implies first flipping bit i and then flipping it back. □

Both properties in Lemma 2 are very useful in our later discussions. We are now in the position to give the construction of a Latin Square by using the basic permutations.

4.1 The First Construction of a Latin Square

Given m basic permutations $\phi_1, \phi_2, \ldots, \phi_m$, we construct a permutation set as follows

$$\Psi = \{\phi_{i_1}\phi_{i_2}\cdots\phi_{i_k} | m \geq i_1 > i_2 > \cdots > i_k \geq 1 \text{ and } 1 \leq k \leq m\}. \quad (8)$$

For example, for $n = 8$, we have

$$\Psi = \{\phi_1, \phi_2, \phi_3, \phi_2\phi_1, \phi_3\phi_1, \phi_3\phi_2, \phi_3\phi_2\phi_1\}.$$

Based on the properties (6) and (7) in Lemma 2, we can see that any composition of one or more basic permutations equals one of the permutations in Ψ. Take the composition $\phi_1\phi_2\phi_1$ as an example. Since

$$\phi_1\phi_2\phi_1 = (\phi_1\phi_2)\phi_1 = (\phi_2\phi_1)\phi_1 = \phi_2(\phi_1\phi_1) = \phi_2 I = \phi_2,$$

this composition equals ϕ_2 which belongs to Ψ.

It is easy to see how many permutations are in set Ψ. In fact, there are $\binom{m}{1}$ permutations which are composed of one basic permutation, $\binom{m}{2}$ permutations which are composed of two basic permutations, and so on. Since

$$\binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{m} = 2^m - 1 = n - 1,$$

it follows that $|\Psi| = n - 1$. Based on the permutation set Ψ, we can construct Latin Squares as described in the following theorem.

Theorem 1. *Let $\rho_1, \rho_2, \ldots, \rho_{n-1}$ be the $n-1$ permutations in Ψ, and $a_0, a_1, \ldots, a_{n-1}$ be a list of numbers such that $\{a_0, a_1, \ldots, a_{n-1}\} = \{0, 1, \ldots, n-1\}$. Then the following matrix is a Latin Square.*

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ \rho_1(a_0) & \rho_1(a_1) & \rho_1(a_2) & \cdots & \rho_1(a_{n-1}) \\ \rho_2(a_0) & \rho_2(a_1) & \rho_2(a_2) & \cdots & \rho_2(a_{n-1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho_{n-1}(a_0) & \rho_{n-1}(a_1) & \rho_{n-1}(a_2) & \cdots & \rho_{n-1}(a_{n-1}) \end{bmatrix}. \quad (9)$$

Proof. See Appendix. □

The above construction of a Latin Square is intuitive and easy to understand, and is especially useful in the proof of the existence of a Latin Square for the self-routable multistage networks in the next section. However, the time complexity of generating a Latin Square this way is high since each permutation in Ψ may contain up to m basic permutations. Although the time complexity of such an off-line and run-once-and-use-forever algorithm is not of our greatest concern, a more practical construction, which takes a minimum possible time effort, is nevertheless appreciated and will be presented in the next subsection.

4.2 The Second Construction of a Latin Square

We first briefly describe the basic idea of our second method. We generate a list of $n-1$ basic permutations (some of which may be identical), and build the Latin Square row by row in an iterative fashion, in the sense that the current row is obtained by applying a basic permutation in the list to the previously generated row. Since there are a total of $n-1$ basic permutations in the list and each of them is applied to n entries of a row, the time complexity of this construction is $O(n^2)$, which is minimum because a Latin Square has n^2 entries. The detailed algorithm is given in Table 2, where *LatinSquare* is the main program, *BuildBasicList* is a recursive function to generate the basic

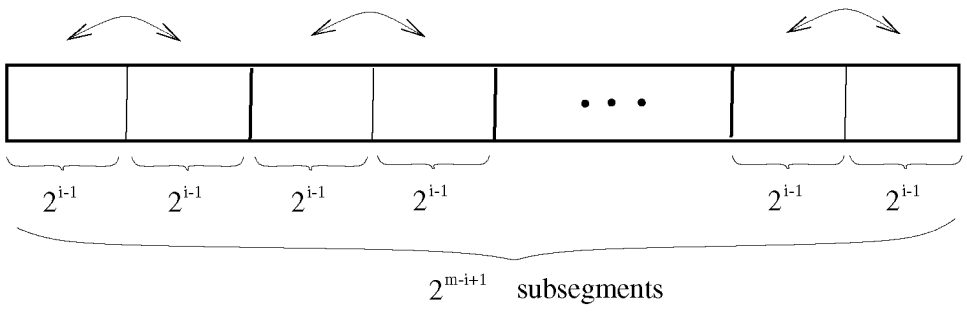

Fig. 5. ϕ_i swaps the two subsegments in a pair.

permutation list, and $BuildLatinSquare$ is the function which actually constructs a Latin Square row by row.

Theorem 2. *The matrix constructed by algorithm LatinSquare in Table 2 is a Latin Square.*

Proof. See Appendix. □

Fig. 6b shows a Latin Square generated by algorithm $LatinSquare$.

Proof. Let the basic permutation list in algorithm $LatinSquare$ be $\{\phi_{k_1}, \phi_{k_2}, \ldots, \phi_{k_{n-1}}\}$. Then the set of permutations which are applied to the original number list $\{a_0, a_1, \ldots, a_{n-1}\}$ in the algorithm is

$$\Psi' = \{\phi_{k_1}, \phi_{k_2}\phi_{k_1}, \phi_{k_3}\phi_{k_2}\phi_{k_1}, \ldots, \phi_{k_{n-1}}\phi_{k_{n-2}}\cdots\phi_{k_2}\phi_{k_1}\}.$$

By Theorem 2, we know that no two permutations in Ψ' are the same, which yields

$$|\Psi'| = n - 1 = |\Psi|.$$

Also, using the properties (6) and (7) of basic permutations, any permutation in Ψ' can be transformed to the format of Ψ in (8). That is,

$$\Psi' = \Psi.$$

Thus, the Latin Squares in Theorem 2 and Theorem 1 are equivalent. □

Now we give an example to illustrate the proof of Theorem 3. For $n = 8$, the basic permutation list is $\{\phi_1, \phi_2, \phi_1, \phi_3, \phi_1, \phi_2, \phi_1\}$. We can list the following one-to-one correspondence between Ψ' and Ψ:

$$\begin{aligned}
\Psi' &= \Psi \\
\phi_1 &= \phi_1 \\
\phi_2\phi_1 &= \phi_2\phi_1 \\
\phi_1\phi_2\phi_1 &= \phi_2 \\
\phi_3\phi_1\phi_2\phi_1 &= \phi_3\phi_2 \\
\phi_1\phi_3\phi_1\phi_2\phi_1 &= \phi_3\phi_2\phi_1 \\
\phi_2\phi_1\phi_3\phi_1\phi_2\phi_1 &= \phi_3\phi_1 \\
\phi_1\phi_2\phi_1\phi_3\phi_1\phi_2\phi_1 &= \phi_3.
\end{aligned}$$

5 Generating Permutations for All-to-All Personalized Exchange in a Class of Multistage Networks

The set of basic permutations ϕ_i $(1 \le i \le m)$ and the Latin Square described in the last section are closely related to the class of self-routable multistage networks under consideration. In fact, we can generate admissible permutations for the class of networks in a generic way to form the Latin Square needed in all-to-all personalized exchange algorithm in Table 1. To do this, we can simply let each stage permutation σ_i (defined in Section 2) be ϕ_1 or I. Recall that ϕ_1 is the permutation $(0,1)(2,3)\ldots(n-2, n-1)$ and I is the identity permutation. Accordingly, all switches in each stage of the network are set to either cross or parallel.

Now we highlight the general approach we will use in this section. We will first prove all such admissible permutations forms a Latin Square by using Theorem 1. Then we will apply the algorithm $LatinSquare$ described in last section to actually construct this Latin Square. Notice that the off-line Latin Square construction algorithm needs to be run only once at the time a network is built.

In the next few subsections, we will show this approach is valid for baseline, omega and indirect binary n-cube networks. In fact, from the method we adopt, one can easily see this approach can be generally applied to the entire class of self-routable multistage networks under consideration.

Finally, it should be pointed out that although C.-L. Wu and T.-Y. Feng [20] proved that baseline, omega, and indirect binary n-cube networks are topologically equivalent, from their proofs one cannot directly find a generic and efficient way to construct the Latin Square for each individual network which is needed in our all-to-all personalized exchange algorithm. Therefore, each network must be discussed separately.

5.1 Baseline Networks

Recall from Section 2, the overall permutation of a baseline network is $\sigma_{m-1}\pi_{m-2}\sigma_{m-2}\ldots\pi_0\sigma_0$, where interstage permutations π_is are defined in (2) and the stage permutations σ_is now take either ϕ_1 or I. We have the following lemmas concerning the properties of the compositions of π_is and ϕ_js.

Lemma 3. *The composition of i $(1 \le i \le m-1)$ consecutive interstage permutations $\pi_{m-2}, \pi_{m-3}, \ldots, \pi_{m-i-1}$ is the following permutation:*

$$p_{m-1}p_{m-2}\ldots p_{i+1}p_i\ldots p_1p_0 \xrightarrow{\pi_{m-2}\pi_{m-3}\cdots\pi_{m-i-1}} p_{m-1}p_{m-2}\ldots p_{i+1}p_0p_1\ldots p_i. \quad (10)$$

Proof. Applying $\pi_{m-i-1}, \pi_{m-i}, \ldots, \pi_{m-3}, \pi_{m-2}$ one by one to a binary number $p_{m-1}p_{m-2}\ldots p_1p_0$, we have

$$\begin{aligned}
&p_{m-1}\ldots p_{i+1}p_i\ldots p_1p_0 \xrightarrow{\pi_{m-i-1}} p_{m-1}\ldots p_{i+1}p_0p_i\ldots \\
&p_1 \xrightarrow{\pi_{m-i}} p_{m-1}\ldots p_{i+1}p_0p_1p_i\ldots \\
&p_3p_2 \xrightarrow{\pi_{m-i+1}} \cdots \xrightarrow{\pi_{m-3}} p_{m-1}\ldots p_{i+1}p_0p_1\ldots \\
&p_{i-2}p_ip_{i-1} \xrightarrow{\pi_{m-2}} p_{m-1}\ldots p_{i+1}p_0p_1\ldots p_{i-1}p_i.
\end{aligned}$$

□

Let

$$\pi = \pi_{m-2}\pi_{m-3}\cdots\pi_1\pi_0, \quad (11)$$

which is the composition of all π_is. π can also be viewed as the overall permutation of a baseline network in which all switches are set to parallel.

The following Corollary gives a special case of Lemma 3, which indicates that π maps a binary number to its reversal.

Corollary 1.

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi} p_0p_1\ldots p_{m-2}p_{m-1}. \quad (12)$$

Corollary 2. *The composition of the i $(1 \le i \le m-1)$ consecutive π_js, ϕ_1, and ϕ_{i+1} satisfies the following equation:*

$$(\pi_{m-2}\pi_{m-3}\cdots\pi_{m-i-1})\phi_1 = \phi_{i+1}(\pi_{m-2}\pi_{m-3}\cdots\pi_{m-i-1}). \quad (13)$$

TABLE 2
The Construction of a Latin Square

```
Algorithm LatinSquare (List {a_0, a_1, ..., a_{n-1}}) /* main */
begin
    List BL ← List {};
    BuildBasicList(m); /* m = log n */
    BuildLatinSquare(BL, {a_0, a_1, ..., a_{n-1}});
end;
Function BuildBasicList(int k)
begin
    if (k == 1)
        BL.append(φ_1);
        return;
    end if
    BuildBasicList(k − 1);
    BL.append(φ_k);
    BuildBasicList(k − 1);
end;
Function BuildLatinSquare(List {φ_{k_1}, φ_{k_2}, ..., φ_{k_{n-1}}}, List {a_0, a_1, ..., a_{n-1}})
begin
    for i = 0 to n − 1 do
        if (i == 0)
            b_0 = a_0; b_1 = a_1; ···; b_{n-1} = a_{n-1};
        else
            b_0 = φ_{k_i}(b_0); b_1 = φ_{k_i}(b_1); ···; b_{n-1} = φ_{k_i}(b_{n-1});
        end if;
        output List {b_0, b_1, ..., b_{n-1}} as one row of the Latin square;
    end for;
end;
```

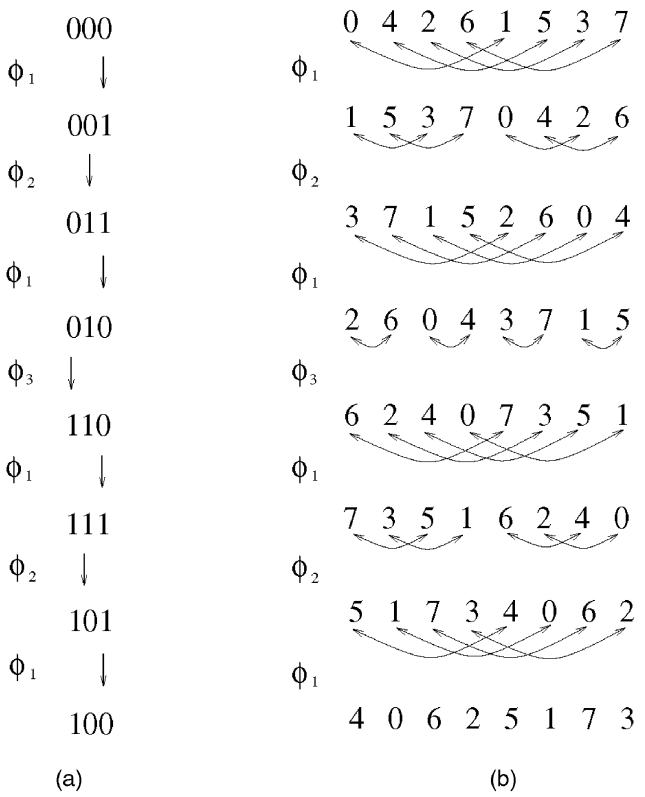

Fig. 6 (a) The 3-bit Gray code sequence generated by applying the basic permutation list to number 0. (b) An 8×8 Latin Square generated by the algorithm in Table 2.

Proof. From Lemma 3 and mapping (5), we can see that the permutations on both sides of (13) map $p_{m-1}p_{m-2}\ldots p_1p_0$ to $p_{m-1}p_{m-2}\ldots p_{i+1}\bar{p}_0p_1\ldots p_i$. □

Theorem 4. *Let the stage permutation of each stage in a baseline network take either ϕ_1 or I (i.e., the switches in this stage are either all set to cross or all set to parallel). The matrix consisting, as its rows, of all the admissible permutations obtained by all possible such switch settings is a Latin Square.*

Proof. Since each stage permutation σ_i takes either ϕ_1 or I, the overall permutation $\sigma_{m-1}\pi_{m-2}\sigma_{m-2}\ \ldots\ \pi_0\sigma_0$ has the following general form for $k \geq 1$ and $0 \leq i_1 < i_2 < \cdots < i_k \leq m-1$:

$$\begin{aligned}&\pi_{m-2}\cdots\pi_{m-i_1-1}\phi_1\pi_{m-i_1-2}\cdots\\&\pi_{m-i_2-1}\phi_1\pi_{m-i_2-2}\cdots\pi_{m-i_k-1}\phi_1\pi_{m-i_k-2}\cdots\pi_1\pi_0\end{aligned} \tag{14}$$

By repeatedly using Corollary 2, we can see that (14) becomes

$$\begin{aligned}&(\phi_{i_1+1}\phi_{i_2+1}\ldots\phi_{i_k+1})(\pi_{m-2}\pi_{m-3}\cdots\pi_1\pi_0)\\&\quad=(\phi_{i_1+1}\phi_{i_2+1}\ldots\phi_{i_k+1})\pi\\&\quad=(\phi_{i_k+1}\phi_{i_{k-1}+1}\ldots\phi_{i_1+1})\pi.\end{aligned}$$

Comparing the set

$$\{\phi_{i_k+1}\phi_{i_{k-1}+1}\ldots\phi_{i_1+1} | k \geq 1, 0 \leq i_1 < i_2 < \cdots < i_k \leq m-1\}$$

with the definition of Ψ in (8), we can see they are exactly the same. Letting

$$a_0 = \pi(0), a_1 = \pi(1), \ldots, a_{n-1} = \pi(n-1),$$

and using Theorem 1, we have that all permutations of form (14) forms a Latin Square. In addition, we can see that π corresponds to the first row of the Latin Square. □

We can actually use algorithm

$$LatinSquare(List\ \{\pi(0), \pi(1), \ldots, \pi(n-1)\})$$

to construct the Latin Square for a baseline network. For example, for an 8×8 network, we first compute by Corollary 1 the first row, $\pi(0), \pi(1), \ldots, \pi(n-1)$, which is $0, 4, 2, 6, 1, 5, 3, 7$, and then call $LatinSquare$ to generate the rest $n-1$ rows of the Latin Square. In Fig. 7, we list all possible switch settings in an 8×8 baseline network, and the corresponding Latin Square is L_1 in (15). L_2 and L_3 in (15) will be used in the next two subsections.

$$L_1 = \begin{bmatrix} 0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \\ 1 & 5 & 3 & 7 & 0 & 4 & 2 & 6 \\ 3 & 7 & 1 & 5 & 2 & 6 & 0 & 4 \\ 2 & 6 & 0 & 4 & 3 & 7 & 1 & 5 \\ 6 & 2 & 4 & 0 & 7 & 3 & 5 & 1 \\ 7 & 3 & 5 & 1 & 6 & 2 & 4 & 0 \\ 5 & 1 & 7 & 3 & 4 & 0 & 6 & 2 \\ 4 & 0 & 6 & 2 & 5 & 1 & 7 & 3 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 3 & 2 & 5 & 4 & 7 & 6 \\ 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\ 2 & 3 & 0 & 1 & 6 & 7 & 4 & 5 \\ 6 & 7 & 4 & 5 & 2 & 3 & 0 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 5 & 4 & 7 & 6 & 1 & 0 & 3 & 2 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \end{bmatrix} \quad (15)$$

$$L_3 = \begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\ 1 & 3 & 5 & 7 & 0 & 2 & 4 & 6 \\ 3 & 1 & 7 & 5 & 2 & 0 & 6 & 4 \\ 2 & 0 & 6 & 4 & 3 & 1 & 7 & 5 \\ 6 & 4 & 2 & 0 & 7 & 5 & 3 & 1 \\ 7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \\ 5 & 7 & 1 & 3 & 4 & 6 & 0 & 2 \\ 4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \end{bmatrix}$$

5.2 Omega Networks

An omega network is depicted in Fig. 2b. As stated in Section 2, The overall permutation of an $n \times n$ omega network is $\sigma_{m-1}\pi_0^{-1}\sigma_{m-2}\pi_0^{-1}\ldots\sigma_1\pi_0^{-1}\sigma_0\pi_0^{-1}$, where the mapping π_0^{-1} is shown in (3). Let π_0^{-i} denote the composition of i permutations π_0^{-1}s. We have the following lemma.

Lemma 4.

$$\pi_0^{-i}\phi_1 = \phi_{i+1}\pi_0^{-i} \text{ for } 0 \le i \le m-1. \quad (16)$$

$$\pi_0^{-m} = I. \quad (17)$$

Proof. When repeatedly applying π_0^{-1} to a binary number $p_{m-1}p_{m-2}\ldots p_1p_0$, we obtain

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi_0^{-1}} p_{m-2}p_{m-3}\ldots$$
$$p_1p_0p_{m-1} \xrightarrow{\pi_0^{-1}} p_{m-3}\ldots p_1p_0p_{m-1}p_{m-2} \xrightarrow{\pi_0^{-1}} \cdots.$$

In general, for $0 \le i \le m-1$, we have

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi_0^{-i}} p_{m-i-1}p_{m-i-2}\ldots p_1p_0p_{m-1}\ldots p_{m-i}. \quad (18)$$

Letting $i = m-1$ and applying π_0^{-1} one more time, we can see that (17) holds. To prove (16), on one hand, we have

$$p_{m-1}p_{m-2}\ldots p_1p_0 \xrightarrow{\pi_0^{-i}\phi_1} p_{m-i-1}p_{m-i-2}\ldots p_1\bar{p}_0p_{m-1}\ldots p_{m-i}.$$

On the other hand, by applying

$$\phi_{i+1}\pi_0^{-i} \text{ to } p_{m-1}p_{m-2}\ldots p_1p_0,$$

we can also obtain

$$p_{m-i-1}p_{m-i-2}\ldots p_1\bar{p}_0p_{m-1}\ldots p_{m-i}.$$

Therefore, equation $\pi_0^{-i}\phi_1 = \phi_{i+1}\pi_0^{-i}$ holds. □

Note that π_0^{-m} is the overall permutation of an omega network with all switches set to parallel. By (17), this permutation is equal to the identity permutation. We can obtain a similar theorem for omega networks to that for baseline networks.

Theorem 5. *Let the stage permutation of each stage in an omega network take either ϕ_1 or I (i.e., the switches in this stage are either all set to cross or all set to parallel). The matrix consisting, as its rows, of all the admissible permutations obtained by all possible such switch settings is a Latin Square.*

Proof. Since each stage permutation σ_i takes either ϕ_1 or I, the overall permutation takes the following format for $k \ge 2$, $i_1 \ge 0$, $i_2, \ldots, i_k \ge 1$, and $i_1 + i_2 + \cdots + i_k = m$

$$\pi_0^{-i_1}\phi_1\pi_0^{-i_2}\phi_1\cdots\pi_0^{-i_{k-1}}\phi_1\pi_0^{-i_k}. \quad (19)$$

By repeatedly using Lemma 4, we can have

$$\begin{aligned}\pi_0^{-i_1}\phi_1\pi_0^{-i_2}\phi_1\cdots\pi_0^{-i_{k-1}}\phi_1\pi_0^{-i_k} &\overset{\text{by(16)}}{=} \phi_{i_1+1}\phi_{(i_1+i_2)+1}\phi_{(i_1+i_2+i_3)+1} \\ &\quad \cdots\phi_{\sum_{j=1}^{k-1} i_j+1}\pi_0^{-m} \\ &\overset{\text{by(17)}}{=} \phi_{i_1+1}\phi_{(i_1+i_2)+1}\phi_{(i_1+i_2+i_3)+1} \\ &\quad \cdots\phi_{\sum_{j=1}^{k-1} i_j+1} \\ &\overset{\text{by(6)}}{=} \phi_{\sum_{j=1}^{k-1} i_j+1}\phi_{\sum_{j=1}^{k-2} i_j+1} \\ &\quad \cdots\phi_{(i_1+i_2)+1}\phi_{i_1+1}\end{aligned}$$

It can be verified that the set

$$\{\phi_{\sum_{j=1}^{k-1} i_j+1}\phi_{\sum_{j=1}^{k-2} i_j+1}\cdots\phi_{(i_1+i_2)+1}\phi_{i_1+1} | k \ge 2,$$
$$i_1 \ge 0, i_2, \ldots, i_k \ge 1, i_1 + i_2 + \cdots + i_k = m\}$$

is equal to Ψ in (8). Letting

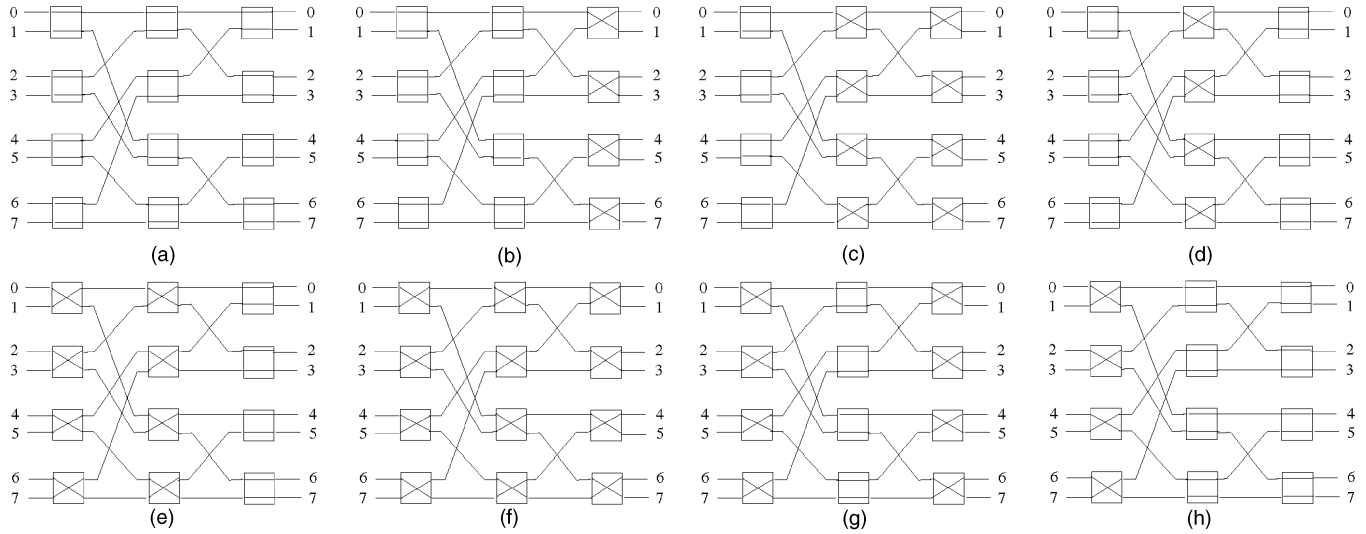

Fig. 7. All possible switch settings, in which each stage is set to either ϕ_1 or I, in an 8×8 baseline network and the corresponding overall permutations realized. (a) III 04261537 (b) IIϕ_1 15370426 (c)$\phi_1\phi_1$I 37152604 (d) Iϕ_1I 26043715 (e) $\phi_1\phi_1$I 62407551 (f) $\phi_1\phi_1\phi_1$ 73516240 (g)ϕ_1Iϕ_1 51734062 (h) ϕ_1II 40625173

$$a_0 = 0, a_1 = 1, \ldots, a_{n-1} = n-1,$$

and using Theorem 1, we can see that all permutations of form (19) forms a Latin Square. □

The generic algorithm

$$LatinSquare(\text{List}\{0, 1, 2, \ldots, n-1\})$$

can be used to construct the Latin Square for an omega network. In Fig. 8, we list all possible switch settings in an 8×8 omega network, and the corresponding Latin Square is L_2 in (15).

5.3 Indirect Binary n-cube Networks

An indirect binary n-cube network is shown in Fig. 2c and as described in Section 2, τ_i, defined in (4) is the interstage permutation between stage i and $i+1$ for $0 \le i \le m-2$ in the network, and the overall permutation of the indirect binary n-cube network is $\sigma_{m-1}\tau_{m-2}\sigma_{m-2}\ldots\tau_0\sigma_0$. Similar to baseline network and omega network, the stage permutations σ_is are now taking either ϕ_1 or I. Let

$$\tau = \tau_{m-2}\tau_{m-3}\ldots\tau_1\tau_0, \tag{20}$$

which is the overall permutation corresponding to that all switches in the network are set to parallel.

Lemma 5. *The composition of i $(1 \le i \le m-1)$ consecutive interstage permutations $\tau_{m-2}, \tau_{m-3}, \ldots, \tau_{m-i-1}$ is the following permutation:*

$$\begin{aligned} &p_{m-1}p_{m-2}\cdots p_{m-i+1}p_{m-i}p_{m-i-1}\cdots p_1p_0 \overset{\tau_{m-2}\tau_{m-3}\cdots\tau_{m-i-1}}{\longrightarrow} \\ &p_{m-2}\cdots p_{m-i+1}p_{m-i}p_0p_{m-i-1}\cdots p_2p_1p_{m-1}. \end{aligned} \tag{21}$$

Proof. Applying $\tau_{m-i-1}, \tau_{m-i}, \ldots, \tau_{m-3}, \tau_{m-2}$ one by one to a binary number $p_{m-1}p_{m-2}\ldots p_1p_0$, we have

$$\begin{aligned} &p_{m-1}\cdots p_{m-i+1}p_{m-i}p_{m-i-1}\cdots p_1p_0 \overset{\tau_{m-i-1}}{\longrightarrow} p_{m-1}\cdots \\ &p_{m-i+1}p_0p_{m-i-1}\cdots p_1p_{m-i} \overset{\tau_{m-i}}{\longrightarrow} p_{m-1}\cdots \\ &p_{m-i+2}p_{m-i}p_0p_{m-i-1}\cdots p_1p_{m-i+1} \overset{\tau_{m-i+1}}{\longrightarrow} \cdots \\ &\overset{\tau_{m-3}}{\longrightarrow} p_{m-1}p_{m-3}\cdots p_{m-i+1}p_{m-i}p_0p_{m-i-1}\cdots p_1p_{m-2} \overset{\tau_{m-2}}{\longrightarrow} p_{m-2}\cdots \\ &p_{m-i+1}p_{m-i}p_0p_{m-i-1}\cdots p_1p_{m-1}. \end{aligned}$$

□

The following Corollary indicates that τ defined in (20) is actually a 1-bit circular-left-shift operation, that is, $\tau = \pi_0^{-1}$.

Corollary 3.

$$p_{m-1}p_{m-2}\ldots p_1p_0 \overset{\tau}{\rightarrow} p_{m-2}\ldots p_2p_1p_0p_{m-1}. \tag{22}$$

Proof. Letting $i = m-1$ in Lemma (5). □

Corollary 4. *The composition of the i $(1 \le i \le m-1)$ consecutive τ_js, ϕ_1 and ϕ_{m-i+1} satisfies the following equation:*

$$\begin{aligned} &(\tau_{m-2}\tau_{m-3}\cdots\tau_{m-i-1})\phi_1 \\ &= \phi_{m-i+1}(\tau_{m-2}\tau_{m-3}\cdots\tau_{m-i-1}). \end{aligned} \tag{23}$$

Proof. By Lemma 5 and the definition of ϕ_i in (5), we can see both permutations map $p_{m-1}p_{m-2}\ldots p_1p_0$ to $p_{m-2}\ldots p_{m-i+1}p_{m-i}\bar{p}_0p_{m-i-1}\ldots p_1p_{m-1}$. □

Theorem 6. *Let the stage permutation of each stage in an indirect binary n-cube network take either ϕ_1 or I (i.e., the switches in this stage are either all set to cross or all set to parallel). The matrix consisting, as its rows, of all the admissible permutations obtained by all possible such switch settings is a Latin Square.*

Proof. Since each stage permutation σ_i takes either ϕ_1 or I, the overall permutation $\sigma_{m-1}\tau_{m-2}\sigma_{m-2}\ \ldots\ \tau_0\sigma_0$ has the following general form for $k \ge 1$ and

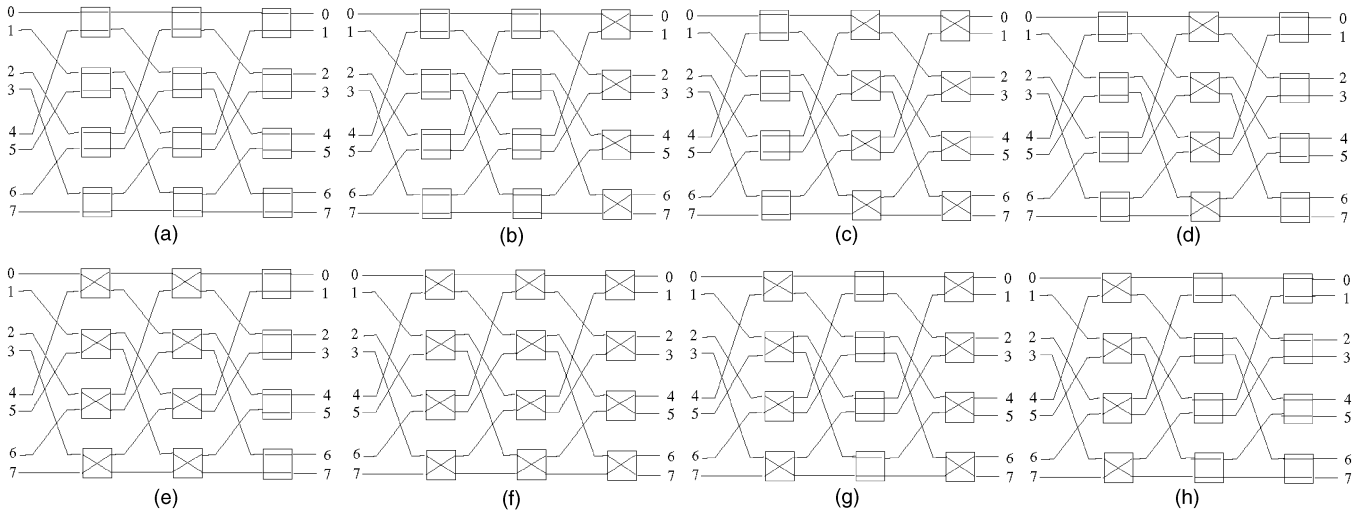

Fig. 8. All possible switch settings, in which each stage is set to either ϕ_1 or I, in an 8×8omega network and the corresponding overall permutations realized. (a) III 01234567 (b) IIϕ_1 10325476 (c) I$\phi_1\phi_1$ 32107654 (d) Iϕ_1I 23016745 (e) $\phi_1\phi_1$I 67452304 (f) $\phi_1\phi_1\phi_1$ 76543210 (g) ϕ_1Iϕ_1 54761032 (h) ϕ_1II 45670123.

$$0 \le i_1 < i_2 < \cdots < i_k \le m-1,$$

$$\begin{aligned}&\tau_{m-2}\ldots\tau_{m-i_1-1}\phi_1\tau_{m-i_1-2}\cdots\\ &\quad\tau_{m-i_2-1}\phi_1\tau_{m-i_2-2}\cdots\tau_{m-i_k-1}\phi_1\tau_{m-i_k-2}\cdots\tau_1\tau_0.\end{aligned} \tag{24}$$

Notice that when $i_1 = 0$, (24) becomes

$$\phi_1\tau_{m-2}\ldots\tau_{m-i_2-1}\phi_1\tau_{m-i_2-2}\cdots\tau_{m-i_k-1}\phi_1\tau_{m-i_k-2}\cdots\tau_1\tau_0.$$

By repeatedly using Corollary 4, (24) becomes

$$\begin{aligned}&(\phi_{m-i_1+1}\phi_{m-i_2+1}\ldots\phi_{m-i_k+1})(\tau_{m-2}\tau_{m-3}\cdots\tau_1\tau_0)\\ &\quad=(\phi_{m-i_1+1}\phi_{m-i_2+1}\ldots\phi_{m-i_k+1})\tau \text{ for } i_1 \ge 1\end{aligned}$$

or

$$\begin{aligned}(\phi_1\phi_{m-i_2+1}\ldots\phi_{m-i_k+1})\tau &= (\phi_{m-i_2+1}\ldots\phi_{m-i_k+1}\phi_1)\tau \text{ for } i_1\\ &= 0.\end{aligned}$$

Comparing the set

$$\begin{aligned}&\{\phi_{m-i_1+1}\phi_{m-i_2+1}\ldots\phi_{m-i_k+1}|k\ge 1,\\ &\ 1\le i_1 i_2\cdots i_k\le m-1\};\bigcup\{\phi_{m-i_2+1}\ldots\\ &\ \phi_{m-i_k+1}\phi_1|k\ge 1, 0=i_1<i_2<\cdots<i_k\le m-1\}\end{aligned}$$

with the definition of Ψ in (8), we know that they are exactly the same. Letting

$$a_0 = \tau(0), a_1 = \tau(1), \ldots, a_{n-1} = \tau(n-1),$$

and using Theorem 1, we can see that all permutations of form (24) forms a Latin Square. Moreover, τ corresponds to the first row of the Latin Square. □

We can use algorithm

$$LatinSquare(\text{List}\{\tau(0), \tau(1), \ldots, \tau(n-1)\})$$

to construct the Latin Square for an indirect binary n-cube network. For example, for an 8×8 network, we first compute by Corollary 3 the first row, $\tau(0), \tau(1), \ldots, \tau(7)$, which is $0, 2, 4, 6, 1, 3, 5, 7$, and then call $LatinSquare$ to generate the rest rows of the Latin Square. In Fig. 9, we list all possible switch settings in an 8×8 indirect binary n-cube network, and the corresponding Latin Square is L_3 n (15).

6 Time Complexity and Comparisons

In this section, we first summarize the time complexity of the proposed all-to-all personalized exchange algorithm for the class of multistage networks. As pointed out in Section 3, the on-line algorithm for an all-to-all personalized exchange takes $O(n + \log n) = O(n)$ time, which matches the lower bound for this type of networks within a constant factor. On the other hand, constructing a Latin Square using the off-line algorithm $LatinSquare$ described in Section 4 takes $O(n^2)$, which is also optimal for this problem. Note that the off-line Latin Square construction algorithm needs to be run only once at the time a network is built, and the Latin Square associated with the network can be viewed as one of the system parameters. Thus, the time complexity of this algorithm is not included in the communication delay.

As this is the first result on all-to-all personalized exchange in self-routable multistage networks, we compare it with existing results in other topologies including hypercube, mesh, and torus networks with both packet switching and wormhole routing. For a meaningful comparison among different topologies which use different switching techniques, we list only asymptotic communication delays here. In fact, due to the fast switch setting in a self-routable multistage network, the delay of passing a switch in a MIN should be shorter than the delay of passing a node in a direct network such as a hypercube, mesh, and torus. However, for an easy comparison, in the following we simply count the delay of a switch in a MIN the same as the delay of a node in a direct network.

In Table 3, we compare the time complexity of our all-to-all personalized exchange algorithm with the existing algorithms for other network topologies, including hypercube and mesh/torus networks. In addi-

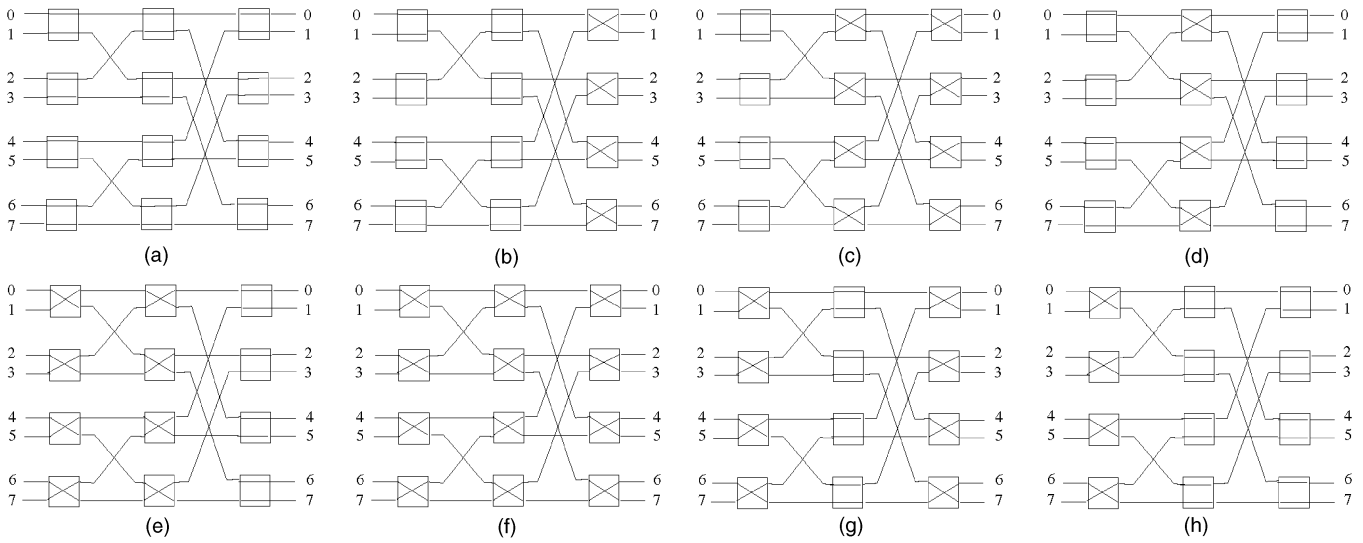

Fig. 9. All possible switch settings, in which each stage of set to either ϕ_1 or I, in an 8×8 indirect binary n-cube network and the corresponding overall permutations realized. (a) III 02461357 (b) IIϕ_1 13570246 (c) I$\phi_1\phi_1$ 57134602 (d) Iϕ_1I 46025713 (e) $\phi_1\phi_1$I 64207531 (f) $\phi_1\phi_1\phi_1$ 75316420 (g) ϕ_1Iϕ_1 31752064 (h) ϕ_1II 20643175.

tion, we also compare the node degree which reflects the number of I/O ports per processor, the number of links used in network which corresponds to network hardware cost, and the diameter which is related to maximum data transmission time.

From Table 3, we can see that the newly proposed algorithm for the class of self-routing multistage networks achieves the minimum time complexity for all-to-all personalized exchange, which is as good as the algorithm for a hypercube in all-port model. On the other hand, in terms of node degree, which reflects the scalability of a network in terms of I/O port limitation, the networks considered in this paper are comparable to (in fact, better than) a mesh or a torus, while a hypercube has a node degree of $\log n$. Thus, a multistage network could be a better choice for implementing all-to-all personalized exchange due to its shorter communication delay and better scalability.

7 Conclusions

In this paper, we have presented an optimal all-to-all personalized exchange algorithm for a class of unique-path, self-routable multistage networks, such as baseline, omega, and indirected binary n cube networks. The new algorithm is based on a special Latin Square, which corresponds to a set of admissible permutations of a multistage network and can be viewed as a system parameter of the network. We have given two methods for constructing the Latin Square used in the algorithm. The first one is useful in the proof of existence of such a Latin Square for each individual multistage network. The second one provides an efficient way to construct the Latin Square. We have also developed a generic method for decomposing all-to-all personalized exchange patterns into admissible permutations to form the Latin Square for the class of multistage networks. The newly proposed algorithm has $O(n)$ time complexity for an $n \times n$ network, which is optimal for all-to-all personalized exchange. By taking advantage of fast setting of self-routable switches and the property of single input/output port per processor in a multistage network, we believe that a multistage network could be a better choice for implementing all-to-all personalized exchange due to its shorter communication latency and better scalability. Our future work will focus on generalizing the idea to other types of MINs and extending the results to optical MINs where some special properties of optics must be considered.

8 Appendix

Proof of Theorem 1. Since

$$\{a_0, a_1, \ldots, a_{n-1}\} = \{0, 1, \ldots, n-1\}$$

and each ρ_i is a permutation, we know that the set of numbers in row i of the matrix,

$$\{\rho_i(a_0), \rho_i(a_1), \ldots, \rho_i(a_{n-1})\} = \{0, 1, \ldots, n-1\}.$$

That is, each row of the matrix forms a permutation.

Now consider the set of numbers in column j of the matrix, $\{a_j, \rho_1(a_j), \rho_2(a_j), \ldots, \rho_{n-1}(a_j)\}$. By the definition of Ψ in (8) and the definition of ϕ_i in (5), if we have some ρ_i, say, $\rho_i = \phi_4\phi_2\phi_1$, then $\rho_i(a_j)$ is the number obtained by flipping bit 4, bit 2, and bit 1 of the binary representation of number a_j. That is, given $i \neq k$, $\rho_i(a_j) \neq \rho_k(a_j)$. Therefore, $\rho_1, \rho_2, \ldots, \rho_{n-1}$ represent distinct non-identity permutations, and $a_j, \rho_1(a_j), \rho_2(a_j), \ldots, \rho_{n-1}(a_j)$ are n distinct numbers which cover all numbers in $\{0, 1, \ldots, n-1\}$. Thus, the column of the matrix also forms a permutation. Hence, the matrix is a Latin Square. □

Proof of Theorem 2. First we need to show that the number of basic permutations generated by function $BuildBasicList(k)$ is $2^k - 1$ so that there are $2^m - 1 = n - 1$ basic permutations in the list passed to function $BuildLatinSquare$ in main program

TABLE 3
Comparisons of Various Networks Used for All-to-All Personalized Exchanges

Network type	Hypercube 1-port model [8]	Hypercube all-port model [8]	2D mesh/torus [11]-[16]	3D mesh/torus [14]-[16]	Baseline, etc. (this paper)
Node degree	$\log n$	$\log n$	4	6	1
Number of links	$n \log n$	$n \log n$	$4n$	$6n$	$n \log n$
Diameter/No. Stages	$\log n$	$\log n$	$O(n^{\frac{1}{2}})$	$O(n^{\frac{1}{3}})$	$\log n$
Communication delay	$O(n \log n)$	$O(n)$	$O(n^{\frac{3}{2}})$	$O(n^{\frac{4}{3}})$	$O(n)$

LatinSquare. Letting this number be $P(k)$, we can establish the following recurrence:

$$P(k) = 2P(k-1) + 1 \text{ and } P(1) = 1 \tag{25}$$

It turns out that the solution to the recurrence (25) is $2^k - 1$.

The algorithm *LatinSquare* generates an $n \times n$ matrix by applying the basic permutation list to the original row, $a_0, a_1, \ldots, a_{n-1}$, in an iterative way. To prove the matrix generated is a Latin Square, let us first focus on the case of applying the basic permutation list to a number a_j iteratively. We apply the first permutation in the list to number a_j and obtain a new number; then apply the second permutation to this new number and obtain another number; and so on. After exhausting all basic permutations in the list, we obtain a list of numbers which form a column of the matrix.

Now lets look at the basic permutation list itself. It actually contains the operations which generate a *Gray code sequence* [24]. A k-bit Gray code sequence contains 2^k binary codewords, each of length k bits, in which two adjacent codewords differ in exactly one bit. For example, for $n = 8$ the basic permutation list is $\{\phi_1, \phi_2, \phi_1, \phi_3, \phi_1, \phi_2, \phi_1\}$. Applying this list to number 0, as shown in Fig. 6a, we obtain

$$\{000, 001, 011, 010, 110, 111, 101, 100\}$$

in binary.

We now generally prove the following claim by induction: Applying the basic permutation list outputed by $BuildBasicList(k)$ to an m-bit binary number, $p_{m-1}p_{m-2}\ldots p_1p_0$, generates a list of all numbers of form

$$p_{m-1}p_{m-2}\ldots p_k\overbrace{\text{xx}\ldots\text{x}}^{k},$$

whose k rightmost bits form a k-bit Gray code sequence.

First, notice that given a binary number b and a basic permutation ϕ_i, b and $\phi_i(b)$ differ only in bit i. When $k = 1$, we have only permutation ϕ_1, and the rightmost bit of all numbers generated, $p_{m-1}p_{m-2}\ldots p_1\text{x}$, form a list $\{0, 1\}$ which is a 1-bit Gray code sequence. Assume the claim holds for $k-1$. Now consider $BuildBasicList(k)$. After the first call of $BuildBasicList(k-1)$, we have a permutation list which can generate all numbers of form

$$p_{m-1}p_{m-2}\ldots p_kp_{k-1}\overbrace{\text{xx}\ldots\text{x}}^{k-1},$$

whose $(k-1)$ rightmost bits form a $(k-1)$-bit Gray code sequence. Then we add ϕ_k to the permutation list. We can apply ϕ_k to the previous number by flipping bit k of the number to obtain a new number. Next, by calling $BuildBasicList(k-1)$ again, we add to the permutation list those permutations which can generate all numbers of form

$$p_{m-1}p_{m-2}\ldots p_k\bar{p}_{k-1}\overbrace{\text{xx}\ldots\text{x}}^{k-1},$$

whose $(k-1)$ rightmost bits form a $(k-1)$-bit Gray code sequence. Thus, the resulting permutation list can generate all numbers of form

$$p_{m-1}p_{m-2}\ldots p_k\overbrace{\text{xx}\ldots\text{x}}^{k},$$

whose k rightmost bits form a k-bit Gray code sequence.

Since in algorithm *LatinSquare*, we apply the basic permutation list outputed by $BuildBasicList(m)$ to any a_j $(0 \le a_j \le n-1)$ in the original row, the resulting number list, which is column j of the matrix obtained by the algorithm, consists of all numbers of an m-bit Gray code sequence which covers $\{0, 1, \ldots, n-1\}$.

On the other hand, the original row $a_0, a_1, \ldots, a_{n-1}$ covers $\{0, 1, \ldots, n-1\}$, so does each of the other $n-1$ rows of the matrix. Hence, the resulting matrix is a Latin Square. □

ACKNOWLEDGMENTS

We would like to thank the anonymous referees of this article for their thoughtful and constructive comments. Y. Yang was supported by the U.S. Army Research Office, under Grant No. DAAH04-96-1-0234 and by the U.S. National Science Foundation, under Grant No. OSR-9350540 and MIP-9522532.

REFERENCES

[1] J. Duato, S. Yalamanchili, and L.M. Ni, *Interconnection Networks: An Engineering Approach*, CS Press, 1997.
[2] D.K. Panda, "Issues in Designing Efficient and Practical Algorithms for Collective Communication on Wormhole-Routed Systems," *Proc. 1995 ICPP Workshop Challenges for Parallel Processing* pp. 8-15, 1995.
[3] Y. Yang and G.M. Masson, "Nonblocking Broadcast Switching Networks," *IEEE Trans. Computers*, vol. 40, no. 9, pp. 1,005-1,015, 1991.
[4] H. Xu, Y. Gui, and L.M. Ni, "Optimal Software Multicast in Wormhole-Routed Multistage Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, No. 6, pp. 597-607, 1997.

[5] V. Varavithya and P. Mohapatra, "Tree-Based Multicasting on Wormhole Routed Multistage Interconnection Networks," *Proc. 1997 Int'l Conf. Parallel Processing* pp. 203-206, 1997.
[6] R. Sivaram, D.K. Panda, and C.B. Stunkel, "Efficient Broadcast and Multicast on Multistage Interconnection Networks Using Multiport Encoding," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 10, pp. 1,004-1,028, Oct. 1998.
[7] Y. Yang, "A Class of Interconnection Networks for Multicasting," *IEEE Trans. Computers,* vol. C-47, no. 8, pp. 899-906, Aug. 1998.
[8] S.L. Johnsson and C.T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers,* vol. 38, no. 9, pp. 1249-1268, Sept. 1989.
[9] Y. Saad, M. H. Schultz, "Data Communication in Parallel Architectures," *Parallel Computing,* vol. 11, pp. 131-150, 1989.
[10] Y. Yang and J. Wang, "Efficient All-to-All Broadcast in All-Port Mesh and Torus Networks," *Proc. Fifth IEEE Int'l Symp. High-Performance Computer Architecture (HPCA-5),* pp. 290-299, Jan. 1999.
[11] D.S. Scott, "Efficient All-to-All Communication Patterns in Hypercube and Mesh Topologies," *Proc. Sixth Distributed Memory Computing Conf.,* pp. 398-403, 1991.
[12] R. Thakur and A. Choudhary, "All-to-All Communication on Meshes with Wormhole Routing," *Proc. Eighth IEEE Int'l Parallel Processing Symp.,* pp. 561-565, Apr. 1994.
[13] Y.-C. Tseng and S. Gupta, "All-to-All Personalized Communication in a Wormhole-Routed Torus," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 5, pp. 498-505, May 1996.
[14] Y.-C. Tseng, T.-H. Lin, S. Gupta, and D.K. Panda, "Bandwidth-Optimal Complete Exchange on Wormhole Routed 2D/3D Torus Networks: A Diagonal-Propagation Approach," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 4, pp. 380-396, Apr. 1997.
[15] Y.J. Suh and S. Yalamanchili, "All-to-All Communication with Minimum Start-up Costs in 2D/3D Tori and Meshes," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 5, pp. 442-458, May 1998.
[16] Y.J. Suh and K.G. Shin, "Efficient All-to-All Personalized Exchange in Multidimensional Torus Networks," *Proc. Int'l Conf. Parallel Processing,* pp. 468-475, Aug. 1998.
[17] D. Gannon and J.V. Rosendale, "On the Impact of Communication Complexity in the Design of Parallel Numerical Algorithms," *IEEE Trans. Computer,* vol. 33, pp. 1,180-1,194, Dec. 1984.
[18] S.L. Johnsson, "Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures," *J. Parallel Distributed Computing,* vol. 4, pp. 133-172, Apr. 1987.
[19] C. Clos, "A Study of Non-Blocking Switching Networks," *The Bell System Technical J.,* vol. 32, pp. 406-424, 1953.
[20] V.E. Benes, "Heuristic Remarks and Mathematical Problems Regarding the Theory of Switching Systems," *The Bell System Technical J.,* vol. 41, pp. 1,201-1,247, 1962.
[21] C.-L. Wu and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. C-29, no. 8, pp. 694-702. Aug. 1980.
[22] L.R. Goke and G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," *Proc. First Ann. Symp. Computer Architecture,* pp. 21-28, 1973.
[23] D. Nassimi and S. Sahni, "A Self-Routing Benes Network and Parallel Permutation Algorithms," *IEEE Trans. Computers,* vol. 30, no. 5, pp. 332-340, May 1981.
[24] K.P. Bogart, *Introductory Combinatorics.* Harcourt Brace Jovanovich, second ed., 1990.
[25] J.P. Hayes, *Computer Architecture and Organization.* McGraw-Hill, 1988.

Yuanyuan Yang received the BEng and MS degrees in computer engineering from Tsinghua University, Beijing, China, in 1982 and 1984, respectively, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland, in 1989 and 1992, respectively. She is currently an associate professor of computer engineering at the State University of New York at Stony Brook. Before joining the State University of New York, she was a faculty member in the department of computer science, University of Vermont in Burlington, from 1992-1999 (as an associate professor from 1998-1999). Her research interests include parallel and distributed computing and systems, high speed networks, optical networks, high performance computer architecture, and fault-tolerant computing. She has published extensively in major journals and refereed conference proceedings related to these research areas. She also holds two U.S. patents in the area of multicast communication networks. Her research has been supported by the U.S. Army Research Office and the U.S. National Science Foundation. She has served on the program/organizing committees of a number of international conferences. She is a senior member of the IEEE and a member of the ACM, IEEE Computer Society, and IEEE Communication Society.

Jianchao Wang is currently a principal member of technical staff at GTE Laboratories Incorporated in Waltham, Massachusetts. He received the BEng degree in computer engineering from Tsinghua University, Beijing, China, in 1982, and the MS and PhD degrees in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1985 and 1988, respectively. Before he joined GTE Labs, he worked at the Institute of Computing Technology, Chinese Academy of Sciences, Johns Hopkins University, and Legent Corporation in Marlboro, Massachusetts. He has received a number of excellence-achievement awards from GTE and Legent, and has five U.S. patents granted or pending. His research interests include IP telephony, databases, programming languages, computer communication networks, computer algorithms, and fault-tolerant computing. He is a member of the IEEE Computer Society and the Association for Computing Machinery.