

# Implementing Load-Balanced Switches With Fat-Tree Networks

Hung-Shih Chueh, Ching-Min Lien, Cheng-Shang Chang, Jay Cheng, and Duan-Shin Lee

Department of Electrical Engineering &

Institute of Communications Engineering

National Tsing Hua University, Hsinchu 30013, Taiwan, R.O.C.

E-mail: d929607@oz.nthu.edu.tw; keiichi@gibbs.ee.nthu.edu.tw;

cschang@ee.nthu.edu.tw; jcheng@ee.nthu.edu.tw; lds@cs.nthu.edu.tw

**Abstract**—Load-balanced switches have received a lot of attention lately as they are much more *scalable* than other existing switch architectures in the literature. One of the most salient features of load-balanced switches is the simplicity of implementing deterministic and periodic connection patterns for its switch fabrics. In this paper, we propose to use *fat-tree* networks as the switch fabrics in load balanced switches. Fat-tree networks have been widely used for interconnecting computers in data centers and for other applications in Network-on-Chip (NoC). One of the main problems in fat-tree networks is that the link capacity has to be increased rapidly from the leaves to the root of the tree. This poses a serious scalability problem as the complexity of implementing the switches near the root of the tree could be very high, especially when a fat-free network is required to be *nonblocking*. As we only require an  $N \times N$  fat-tree network to realize a set of  $N$  permutations needed for the implementation of  $N \times N$  load-balanced switches, in this paper we show that the implementation complexity can be greatly reduced. For this, we first derive a lower bound on the link capacity for each switch in a fat-tree network. By using the *uniform mapping property* of the *bit-reversal* permutation, we show that there exists a set of  $N$  permutations that achieves the lower bound. To further reduce the implementation complexity, we propose a fully meshed fat-tree network that replaces the upper half of the tree by a simple mesh. We then show a fully meshed fat-tree network can be implemented by folding a banyan-type network that realizes this set of  $N$  permutations.

## I. INTRODUCTION

Load-balanced switches [1]–[8] have received a lot of attention recently as they are much more *scalable* than other existing switch architectures in the literature. In Figure 1, we show a typical two-stage load-balanced switch: the first stage is for load-balancing that converts incoming traffic into the uniform traffic, and the second stage is for switching of the uniform traffic. It was shown in [1] that load-balanced switches achieve 100% throughput and have delay performance comparable to ideal output-buffered switches (when the traffic is heavy and bursty).

In this paper, we consider the discrete-time setting and assume that time is slotted and synchronized so that a fixed-size packet can be transmitted over a link within a time slot. One of the most salient features of load-balanced switches is that the connection patterns for the switch fabrics in both stages in Figure 1 are *deterministic* and *periodic*. As such,

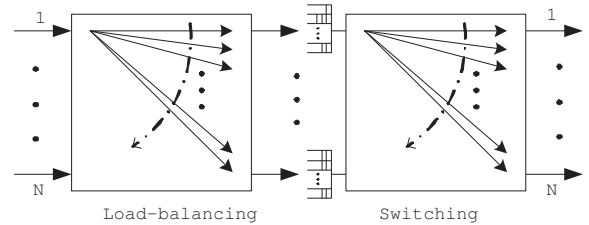


Fig. 1. The generic two-stage load-balanced switch.

there is no need to find matchings as required in most input-buffered switches, and there are no computational overheads in load-balanced switches. Specifically, for an  $N \times N$  load-balanced switch, the switch fabrics in both stages in Figure 1 only need to realize in every period of  $N$  time slots any  $N \times N$  permutation matrices  $P_1, P_2, \dots, P_N$  that satisfy

$$P_1 + P_2 + \dots + P_N = \mathbf{e}, \quad (1)$$

where  $\mathbf{e}$  is the  $N \times N$  matrix with all its elements being 1. Clearly, there are many possible choices of the  $N$  permutation matrices  $P_1, P_2, \dots, P_N$  for implementing  $N \times N$  load-balanced switches. In the literature, there are two well-known conditionally nonblocking switches, i.e., rotators (which realize all of the powers of the circular shift matrix) and symmetric TDM switches [7]–[8], that can be used for realizing the needed connection patterns for the switch fabrics in load balanced switches. In [7] and [8], respectively, it was shown that twister networks (which are special types of multistage interconnection networks) and degenerate banyan networks (which only use half of the inputs/outputs in the classical banyan networks) can be used as rotators and symmetric TDM switches, and hence they can be used as the switch fabrics in load balanced switches.

In this paper, we propose to use *fat-tree* networks [9] as the switch fabrics in load balanced switches. Fat-tree networks (and many variants of them) are commonly used for the constructions of data center interconnection networks [10]. They have also been widely deployed in Network-on-Chip (NoC) [11]. As suggested by its name, a fat-tree network is a switching network constructed from a complete binary tree,

where each leaf is an input/output port and each non-leaf node is a switch. To accommodate the traffic multiplexed into the tree, the link capacity of a switch has to be increased rapidly from the leaves to the root of the tree (this is why it is called a fat-tree network). In particular, if a fat-tree network with  $N$  input/output ports is required to be *nonblocking*, i.e., the fat-free network can realize all of the  $N!$  permutations between its input and output ports, then each node has to be a nonblocking switch and the link capacity of a switch has to be increased *exponentially* from the bottom to the top of the tree. This poses a serious scalability issue in designing the switches near the root of the tree when  $N$  is very large. Fortunately, as our purpose in this paper is to use fat-tree networks as the switch fabrics in load balanced switches, we only require a fat-tree network with  $N$  input/output ports to realize  $N$  permutation matrices  $P_1, P_2, \dots, P_N$  that satisfy the condition in (1) in every period of  $N$  time slots. We will show that the scalability problem in a nonblocking fat-tree network can be solved in this scenario by appropriate choices of the  $N$  permutation matrices  $P_1, P_2, \dots, P_N$  and by replacing the upper half of the fat-tree network by a simple mesh.

In this paper, we first show a lower bound on the link capacity for each switch in a fat-tree network that is capable of realizing any  $N$  permutation matrices  $P_1, P_2, \dots, P_N$  satisfying the condition in (1). The lower bound is derived based on averaging the traffic flows that need to go through a switch in the fat-tree network. Unfortunately, both rotators and symmetric TDM switches require link capacities that are substantially higher than those given by the lower bound.

Then we propose new permutation matrices  $P_1, P_2, \dots, P_N$  that not only satisfy the condition in (1) but also achieve the lower bound, i.e., they can be realized by a fat-tree network with link capacities specified by the lower bound. The idea is based on the *bit-reversal* permutation previously proposed in [12]. One key property of the bit-reversal permutation is the *uniform mapping property* that maps the inputs in a subtree uniformly to the outputs in other subtrees. We show that any permutation that has the uniform mapping property can be realized by a fat-tree network with link capacities specified by the lower bound. We also show that the bit-reversal permutation and its variants obtained by circular shifts all have the uniform mapping property, and hence they can be realized by a fat-tree network with link capacities specified by the lower bound.

To further reduce the implementation complexity, we propose a *fully meshed* fat-tree network by replacing the upper half of a fat-tree network by a simple mesh. As such, there is no need to implement the switches in the upper half of the usual fat-tree network, and hence we do not have the scalability problem for the switches near the root in the usual nonblocking fat-tree network. We show that any permutation that has the uniform mapping property can also be realized by a fully meshed fat-tree network.

To implement the switches in a fully meshed fat-tree network, i.e., in the lower half of a fat-tree network, we consider a specific banyan-type network. Such a banyan-type

network is constructed by connecting a collection of reversed baseline networks (with a much smaller size) and another collection of baseline networks (with a much smaller size). We show that a permutation can be realized by such a banyan-type network if and only if the permutation has the uniform mapping property. By folding such a banyan-type network, we are able to map such a folded banyan-type network to the corresponding fully meshed fat-tree network with the same number of input/output ports. We further show that each switch in the corresponding fully meshed fat-tree network can be constructed by a collection of  $2 \times 2$  switches in the folded banyan-type network. As any banyan-type network possesses the *self-routing* property, the corresponding fully meshed fat-tree network also inherits such a self-routing property. This solves both the switch implementation problem and the routing problem in a fully meshed fat-tree network.

This rest of this report is organized as follows. In Section II, we introduce fat-tree networks and show a lower bound on the link capacity for each switch in a fat-tree network that can realize the permutations needed for load-balanced switches. In Section III, we introduce the bit-reversal permutation and show that any permutation that has the uniform mapping property, including the bit-reversal permutation and its variants obtained by circular shifts, can be realized by a fat-tree network with the link capacities specified by the lower bound. We then propose fully meshed fat-tree networks in Section IV and show that a fully meshed fat-tree network can be implemented by folding a specific banyan-type network. We also show that any permutation that has the uniform mapping property can be realized by a fully meshed fat-tree network with the link capacities specified by the lower bound. Finally, the report is concluded in Section VI, where we address possible extensions of our work.

## II. FAT-TREE NETWORKS

A fat-tree network, first proposed in [9], is a switching network constructed from a complete binary tree. To explain how a fat-tree network works, we first consider a complete binary tree with  $2^n$  leaves, indexed from 0 to  $2^n - 1$  (see Figure 2 for a complete binary tree with 16 leaves). In such a complete binary tree, there are  $n + 1$  levels, indexed from 0 to  $n$ . The root is the only node at level 0 and a node is at level  $j + 1$  if it is a child of a node at level  $j$  for  $0 \leq j \leq n - 1$ . Index the root as node  $(0, 0)$ , and recursively index the two children of node  $(j, k)$  as node  $(j + 1, 2k)$  and node  $(j + 1, 2k + 1)$  for  $0 \leq j \leq n - 1$  and  $0 \leq k \leq 2^j - 1$ . Clearly, there are  $2^j$  nodes at level  $j$  and in this report we also call node  $(j, k)$  as the  $k^{\text{th}}$  node at level  $j$  for  $0 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ . Note that the  $2^n$  leaves are nodes  $(n, 0), (n, 1), \dots, (n, 2^n - 1)$ , and we also call node  $(n, x)$  as leaf  $x$  for  $0 \leq x \leq 2^n - 1$  in this report.

For  $0 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ , let  $T(j, k)$  be the subtree rooted at node  $(j, k)$ , and let  $S(j, k)$  be the set of all of the leaves in the subtree  $T(j, k)$ , namely,

$$S(j, k) = \{x : k \cdot 2^{n-j} \leq x \leq (k + 1)2^{n-j} - 1\} \quad (2)$$

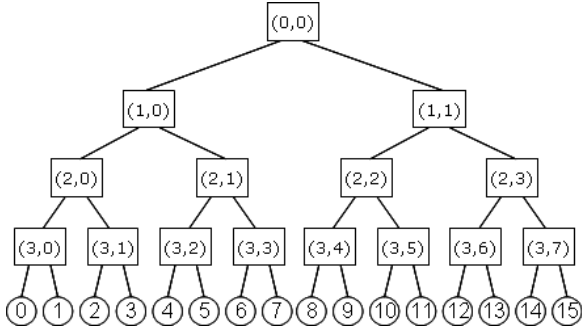


Fig. 2. A complete binary tree with 16 leaves.

Note that the total number of leaves in the subtree  $T(j, k)$  is  $|S(j, k)| = 2^{n-j}$ . For example, for the complete binary tree with 16 leaves in Figure 2, we have  $S(3, 2) = \{4, 5\}$ ,  $S(3, 3) = \{6, 7\}$ , and  $S(2, 1) = S(3, 2) \cup S(3, 3) = \{4, 5, 6, 7\}$ . For  $0 \leq x \leq 2^n - 1$ , let the  $n$ -tuple  $(I_n(x), I_{n-1}(x), \dots, I_1(x))$  be the binary representation of  $x$ , i.e.,  $x = \sum_{m=1}^n I_m(x)2^{m-1}$ , where  $I_m(x)$  is the  $(n-m+1)$ <sup>th</sup> most significant bit of  $x$  for  $1 \leq m \leq n$ . Then the set  $S(j, k)$  can be alternatively expressed as

$$S(j, k) = \{x : 0 \leq x \leq 2^n - 1, \text{ and } I_{n-j+m}(x) = I_m(k) \text{ for } 1 \leq m \leq j\}. \quad (3)$$

In other words,  $S(j, k)$  contains all of the leaves of which the first  $j$  most significant bits are the same as the last  $j$  most significant bits of  $k$  (note that the first  $n-j$  most significant bits of  $k$  are 0).

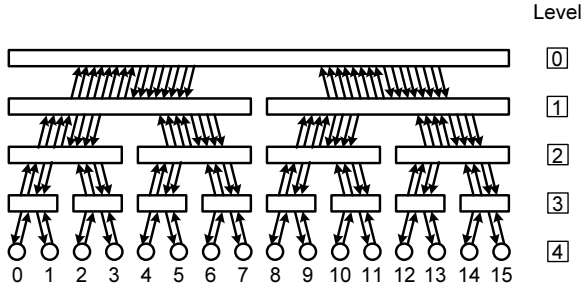


Fig. 3. A nonblocking fat-tree network with 16 input/output ports.

Now we show how one constructs a  $2^n \times 2^n$  nonblocking fat-tree network (with  $2^n$  input/output ports) by using the complete binary tree with  $2^n$  leaves. For this, we view every leaf of the tree as both an input port and an output port of a  $2^n \times 2^n$  switching network and view every non-leaf node in the tree as a nonblocking switch (see Figure 3 for a nonblocking fat-tree network with 16 input/output ports). For  $1 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ , let the upward capacity  $C_u(j, k)$  of node  $(j, k)$  be the number of *parallel links* from node  $(j, k)$  to its parent. For  $0 \leq j \leq n-1$  and  $0 \leq k \leq 2^j - 1$ , let the downward capacity  $C_d(j, k)$  of node  $(j, k)$  be the number of *parallel links* from node  $(j, k)$  to its two children. In this report, we assume that the downward capacity of a node is *evenly split* between its two children.

For such a  $2^n \times 2^n$  fat-tree network to be nonblocking, it has to realize all of the  $(2^n)! 2^n \times 2^n$  permutations between its input and output ports. In other words, for each  $2^n \times 2^n$  permutation there is a *non-conflicting* path for every pair of input/output ports specified by the permutation. Clearly, this requires that the upward capacity and downward capacity of a node in the tree must not be less than the total number of leaves in the subtree rooted at this node. As the total number of leaves in the subtree  $T(j, k)$  is  $|S(j, k)| = 2^{n-j}$  for  $0 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ , we have

$$C_u(j, k) \geq 2^{n-j}, \quad 1 \leq j \leq n, \quad 0 \leq k \leq 2^j - 1, \quad (4)$$

$$C_d(j, k) \geq 2^{n-j}, \quad 0 \leq j \leq n-1, \quad 0 \leq k \leq 2^j - 1. \quad (5)$$

Conversely, if the upward capacities and the downward capacities of the nodes in a fat-tree network satisfy the conditions in (4) and (5), respectively, then there is always a non-conflicting path from an input to the root and there is always a non-conflicting path from the root to an output, and it follows that the fat-tree network is nonblocking. Therefore, the conditions in (4) and (5) are the necessary and sufficient conditions for a fat-tree network to be nonblocking. In this report, we define a  $2^n \times 2^n$  nonblocking fat-tree network to be the fat-tree network with  $C_u(j, k) = 2^{n-j}$  for  $1 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ , and  $C_d(j, k) = 2^{n-j}$  for  $0 \leq j \leq n-1$  and  $0 \leq k \leq 2^j - 1$ . Note that the link capacity of a switch in a nonblocking fat-tree network grows exponentially from the leaves to the root, and this poses a serious scalability problem in designing the switches near the root.

As mentioned in Section I, our purpose in this report is to use fat-tree networks as the switch fabrics in load balanced switches, and hence we only require a  $2^n \times 2^n$  fat-tree network to realize  $2^n$  permutation matrices  $P_1, P_2, \dots, P_{2^n}$  that satisfy the condition in (1) in every period of  $2^n$  time slots. As such, it seems that the upward capacities and the downward capacities of the nodes in such a fat-tree network could be greatly reduced. In the following theorem, we first show lower bounds for the upward capacities and the downward capacities of the nodes in such a fat-tree network.

**Theorem 1** *Suppose that a  $2^n \times 2^n$  fat-tree network is capable of realizing  $2^n$  permutation matrices  $P_1, P_2, \dots, P_{2^n}$  that satisfy the condition in (1) in every period of  $2^n$  time slots. Then for each  $0 \leq k \leq 2^j - 1$ , we have*

$$C_u(j, k) \geq \begin{cases} 2^{n-j} - 2^{n-2j} \\ 2^{n-j} - 2^{n-2j}, \text{ if } 1 \leq j \leq \lfloor n/2 \rfloor, \\ 2^{n-j}, \text{ if } \lfloor n/2 \rfloor + 1 \leq j \leq n, \end{cases} \quad (6)$$

$$C_d(j, k) \geq 2 \begin{cases} 2^{n-j-1} - 2^{n-2j-2} \\ 2^{n-j} - 2^{n-2j-1}, \text{ if } 0 \leq j \leq \lfloor n/2 \rfloor - 1, \\ 2^{n-j}, \text{ if } \lfloor n/2 \rfloor \leq j \leq n-1. \end{cases} \quad (7)$$

**Proof.** Consider node  $(j, k)$ , where  $0 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$ . Our idea for the proof of (6) and (7) is by *averaging*. Consider a frame of  $2^n$  time slots, indexed from 1 to  $2^n$ , called the tagged frame. Assume that there is always a packet

at every input port in every time slot and assume that fat-tree network realizes permutation matrix  $P_i$  in the  $i^{\text{th}}$  time slot for  $i = 1, 2, \dots, 2^n$ . Since the sum of the  $2^n$  permutation matrices  $P_1, P_2, \dots, P_{2^n}$  is a  $2^n \times 2^n$  matrix with all its elements being 1, there is exactly one packet transmitted from every input to every output in the tagged frame.

For  $1 \leq j \leq n$ , every leaf in the subtree  $T(j, k)$  sends a packet to every leaf that is *not* in the subtree  $T(j, k)$  in the tagged frame, and each of these packets has to go through an upward link of node  $(j, k)$ . As there are  $2^{n-j}$  leaves in the subtree  $T(j, k)$  and there are  $2^n - 2^{n-j}$  leaves that are not in the subtree  $T(j, k)$ , the total number of packets that have to go through the upward links of node  $(j, k)$  in the tagged frame is  $2^{n-j}(2^n - 2^{n-j})$ . On the average, there are  $2^{n-j} - 2^{n-2j}$  packets that need to go through the upward links of node  $(j, k)$  per time slot. Clearly, the upward capacity  $C_u(j, k)$  of node  $(j, k)$  is not less than the maximum number of packets that need to go through its upward links in a time slot, and hence is also not less than the average number of packets that need to go through its upward links per time slot. Therefore, it follows that  $C_u(j, k) \geq \lceil 2^{n-j} - 2^{n-2j} \rceil$  for  $1 \leq j \leq n$ , which is the desired result in (6).

For  $0 \leq j \leq n-1$ , every leaf that is *not* in the subtree  $T(j+1, 2k)$  (resp., subtree  $T(j+1, 2k+1)$ ) sends a packet to every leaf in the subtree  $T(j+1, 2k)$  (resp., subtree  $T(j+1, 2k+1)$ ) in the tagged frame, and each of these packets has to go through a downward link of node  $(j, k)$  that is directed to node  $(j+1, 2k)$  (resp., node  $(j+1, 2k+1)$ ). On the average, there are  $2^{n-j-1} - 2^{n-2j-2}$  packets that need to go through the downward links of node  $(j, k)$  per time slot that are directed to node  $(j+1, 2k)$  (resp., node  $(j+1, 2k+1)$ ). As we assume that the downward capacity of a node is evenly split between its two children, we then see that  $C_d(j, k) \geq 2 \lceil 2^{n-j-1} - 2^{n-2j-2} \rceil$  for  $0 \leq j \leq n-1$ , which is the desired result in (7). ■

Observe that lower bounds for the upward capacities and the downward capacities in (6) and (7) are the same as those of nonblocking fat-tree networks in (4) and (5) for the lower half of the tree, but are smaller for the upper half of the tree. As the scalability problem is mainly due to the design of the switches in the upper half of the tree, it is of highly importance and interest to see if there exist  $2^n$  permutation matrices  $P_1, P_2, \dots, P_{2^n}$  that satisfy the condition in (1) and achieve the lower bounds in (6) and (7). It can be seen that the  $2^n$  permutation matrices realized by rotators and symmetric TDM switches do not achieve the lower bounds in (6) and (7). In the next section, we will find  $2^n$  permutation matrices  $P_1, P_2, \dots, P_{2^n}$  that satisfy the condition in (1) and achieve the lower bounds in (6) and (7).

### III. BIT-REVERSAL PERMUTATION

In the previous section, we derive lower bounds for the upward capacities and the downward capacities of a fat-tree network to realize the needed permutations for a load-balanced switch. In this section, we will show that these lower bounds

are indeed achievable by using the bit-reversal permutation introduced in [12] and its variants obtained by circular shifts.

We first introduce some notations that will be used for describing the bit-reversal permutation. Let  $Z_N = \{0, 1, \dots, N-1\}$ . For a permutation  $\sigma$  on  $Z_N$ , we denote  $P_\sigma$  as the  $N \times N$  permutation matrix corresponding to  $\sigma$ . For a set  $S \subseteq Z_N$ , let  $\sigma(S)$  be the range of  $S$  under  $\sigma$ , i.e.,  $\sigma(S) = \{\sigma(x) : x \in S\}$ .

Let  $\sigma_c$  be the circular shift permutation on  $Z_N$ , i.e.,  $\sigma_c(x) = (x+1) \bmod N$  for all  $0 \leq x \leq N-1$ . Also, let  $\sigma_c^i$  be the identity permutation on  $Z_N$  for  $i=0$  and let  $\sigma_c^i = \sigma_c^{i-1} \circ \sigma_c$  for  $i \geq 1$ . Clearly,  $\sigma_c^i$  is the permutation that performs circular shift permutation  $i$  times for  $i \geq 0$ , i.e.,  $\sigma_c^i(x) = (x+i) \bmod N$  for  $0 \leq x \leq N-1$ . As it is easy to see that

$$P_{\sigma_c^0} + P_{\sigma_c^1} + \dots + P_{\sigma_c^{N-1}} = \mathbf{e}, \quad (8)$$

we have that  $P_{\sigma_c^0}, P_{\sigma_c^1}, \dots, P_{\sigma_c^{N-1}}$  satisfy the condition in (1).

Furthermore, for a permutation  $\sigma$  on  $Z_N$ , we denote  $\sigma_i = \sigma_c^i \circ \sigma$  for  $i \geq 0$ , i.e.,  $\sigma_i(x) = \sigma_c^i(\sigma(x)) = (\sigma(x) + i) \bmod N$  for  $0 \leq x \leq N-1$ . Since  $P_{\sigma_i} = P_{\sigma_c^i} P_\sigma$  for  $i \geq 0$ , we see from (8) that

$$\begin{aligned} P_{\sigma_0} + P_{\sigma_1} + \dots + P_{\sigma_{N-1}} &= (P_{\sigma_c^0} + P_{\sigma_c^1} + \dots + P_{\sigma_c^{N-1}}) P_\sigma \\ &= \mathbf{e} P_\sigma = \mathbf{e}. \end{aligned} \quad (9)$$

Note that as  $\sigma_c^0$  is the identity permutation on  $Z_N$ , we have  $\sigma_0 = \sigma_c^0 \circ \sigma = \sigma$ . Therefore, it follows from (9) that  $P_\sigma, P_{\sigma_1}, P_{\sigma_2}, \dots, P_{\sigma_{N-1}}$  satisfy the condition in (1).

**Definition 2 (Bit-Reversal Permutation)** Let  $N = 2^n$  and let  $(I_n(x), I_{n-1}(x), \dots, I_1(x))$  be the binary representation of  $x \in Z_N$ , where  $I_m(x)$  is the  $(n-m+1)^{\text{th}}$  most significant bit of  $x$  for  $1 \leq m \leq n$ . The bit-reversal permutation  $\pi$  on  $Z_N$  is the permutation such that

$$I_m(\pi(x)) = I_{n+1-m}(x), \text{ for } 1 \leq m \leq n, \quad (10)$$

namely,  $\pi(x) = \sum_{m=1}^n I_{n+1-m}(x) 2^{m-1}$ , for  $0 \leq x \leq N-1$ .

Note that  $P_\pi, P_{\pi_1}, P_{\pi_2}, \dots, P_{\pi_{N-1}}$  satisfy the condition in (1), where  $\pi_i = \sigma_c^i \circ \pi$  for  $1 \leq i \leq N-1$ . In Figure 4, we show the 16 permutations  $\pi, \pi_1, \dots, \pi_{15}$  on  $Z_{16}$ . The column marked with 0 on the top row is  $\pi$ , and the column marked with  $i$  on the top row is  $\pi_i$  for  $1 \leq i \leq 15$ . As  $\pi, \pi_1, \dots, \pi_{15}$  satisfy the condition in (1), the  $16 \times 16$  matrix in Figure 4 is a Latin square, where every symbol in  $Z_{16} = \{0, 1, 2, \dots, 15\}$  appears exactly once in every row and every column.

One prominent property of the bit-reversal permutation is the uniform mapping property as defined below.

**Definition 3 (Uniform Mapping Property)** Let  $N = 2^n$ . A permutation  $\sigma$  on  $Z_N$  is said to have the uniform mapping property if

$$|\sigma(S(j, k)) \cap S(n-j, \ell)| = 1 \quad (11)$$

for all  $0 \leq j \leq n$ ,  $0 \leq k \leq 2^j - 1$ , and  $0 \leq \ell \leq 2^{n-j} - 1$ , where  $S(j, k)$  and  $S(n-j, \ell)$  are given by (3).



*shortest paths* for all of the pairs of input/output ports specified by  $\sigma$  are non-conflicting paths. The shortest path from an input port  $x$  to its output port  $\sigma(x)$  is given by first going up the tree from  $x$  to the first common ancestor of  $x$  and  $\sigma(x)$ , and then going down the tree to  $\sigma(x)$ .

(i) We first show that there is no conflict in the upward links of node  $(j, k)$  for  $1 \leq j \leq n$  and  $0 \leq k \leq 2^j - 1$  by proving that  $C_u(j, k)$  is not less than the total number of shortest paths that go through its upward links. We consider the two cases  $1 \leq j \leq \lfloor n/2 \rfloor$  and  $\lfloor n/2 \rfloor + 1 \leq j \leq n$  separately.

*Case 1:*  $1 \leq j \leq \lfloor n/2 \rfloor$ . Note that a shortest path needs to go through an upward link of node  $(j, k)$  if its input  $x$  is a leaf of the subtree  $T(j, k)$  and its output  $\sigma(x)$  is a leaf outside the subtree  $T(j, k)$ . The set of the leaves that are outside the subtree  $T(j, k)$  can be written as  $\cup_{k' \neq k} S(j, k')$ . Since in this case we have  $j \leq \lfloor n/2 \rfloor$ , the set of the leaves in the subtree  $T(j, k')$  can be expressed as the union of the sets of the leaves in the subtrees  $T(n-j, \ell)$ ,  $\ell = 2^{n-2j}k', 2^{n-2j}k' + 1, \dots, 2^{n-2j}(k'+1) - 1$ , and hence we have  $S(j, k') = \cup_{\ell=2^{n-2j}k'}^{2^{n-2j}(k'+1)-1} S(n-j, \ell)$ .

Therefore, it follows from the uniform mapping property in (11) and (6) that the total number of shortest paths that go through the upward links of node  $(j, k)$  is given by

$$\begin{aligned}
& |\sigma(S(j, k)) \cap (\cup_{k' \neq k} S(j, k'))| \\
&= |\sigma(S(j, k)) \cap (\cup_{k' \neq k} \cup_{\ell=2^{n-2j}k'}^{2^{n-2j}(k'+1)-1} S(n-j, \ell))| \\
&= |\cup_{k' \neq k} \cup_{\ell=2^{n-2j}k'}^{2^{n-2j}(k'+1)-1} (\sigma(S(j, k)) \cap S(n-j, \ell))| \\
&= \sum_{k' \neq k} \sum_{\ell=2^{n-2j}k'}^{2^{n-2j}(k'+1)-1} |\sigma(S(j, k)) \cap S(n-j, \ell)| \\
&= \sum_{k' \neq k} \sum_{\ell=2^{n-2j}k'}^{2^{n-2j}(k'+1)-1} 1 = (2^j - 1)2^{n-2j} \\
&= C_u(j, k). \tag{18}
\end{aligned}$$

*Case 2:*  $\lfloor n/2 \rfloor + 1 \leq j \leq n$ . Clearly, the total number of shortest paths that go through the upward links of node  $(j, k)$  is bounded above by  $|S(j, k)|$ , i.e., the total number of leaves in the subtree  $T(j, k)$ . As in this case we have from (6) that  $C_u(j, k) = 2^{n-j} = |S(j, k)|$ , the proof is completed.

(ii) Now we show that there is no conflict in the downward links of node  $(j, k)$  for  $0 \leq j \leq n-1$  and  $0 \leq k \leq 2^j - 1$  by proving that  $C_d(j, k)$  is not less than the total number of shortest paths that go through its downward links. We consider the two cases  $0 \leq j \leq \lfloor n/2 \rfloor - 1$  and  $\lfloor n/2 \rfloor \leq j \leq n-1$  separately.

*Case 1:*  $0 \leq j \leq \lfloor n/2 \rfloor - 1$ . Note that a shortest path needs to go through a downward link of node  $(j, k)$  that is directed to node  $(j+1, 2k)$  (resp., node  $(j+1, 2k+1)$ ) if its input  $x$  is a leaf outside the subtree  $T(j+1, 2k)$  (resp., subtree  $T(j+1, 2k+1)$ ) and its output  $\sigma(x)$  is a leaf of the subtree  $T(j+1, 2k)$  (resp., subtree  $T(j+1, 2k+1)$ ). The set of the leaves that are outside the subtree  $T(j+1, 2k)$  (resp., subtree  $T(j+1, 2k+1)$ ) can be written as  $\cup_{k' \neq 2k} S(j+1, k')$  (resp.,  $\cup_{k' \neq 2k+1} S(j+1, k')$ ). Since in this case we have  $j +$

$1 \leq \lfloor n/2 \rfloor$ , the set of the leaves in the subtree  $T(j+1, k')$  can be expressed as the union of the sets of the leaves in the subtrees  $T(n-j-1, \ell)$ ,  $\ell = 2^{n-2j-2}k', 2^{n-2j-2}k' + 1, \dots, 2^{n-2j-2}(k'+1) - 1$ , and hence we have  $S(j+1, k') = \cup_{\ell=2^{n-2j-2}k'}^{2^{n-2j-2}(k'+1)-1} S(n-j-1, \ell)$ .

As such, it follows from the uniform mapping property in (11) and (7) that the total number of shortest paths that go through the downward links of node  $(j, k)$  that are directed to node  $(j+1, 2k)$  (resp., node  $(j+1, 2k+1)$ ) is given by

$$\begin{aligned}
& |\sigma(\cup_{k' \neq 2k} S(j+1, k')) \cap S(j+1, 2k)| \\
&= |\sigma(\cup_{k' \neq 2k} \cup_{\ell=2^{n-2j-2}k'}^{2^{n-2j-2}(k'+1)-1} S(n-j-1, \ell) \\
&\quad \cap S(j+1, 2k))| \\
&= |\cup_{k' \neq 2k} \cup_{\ell=2^{n-2j-2}k'}^{2^{n-2j-2}(k'+1)-1} \sigma(S(n-j-1, \ell) \\
&\quad \cap S(j+1, 2k))| \\
&= \sum_{k' \neq 2k} \sum_{\ell=2^{n-2j-2}k'}^{2^{n-2j-2}(k'+1)-1} |\sigma(S(n-j-1, \ell) \\
&\quad \cap S(j+1, 2k))| \\
&= \sum_{k' \neq 2k} \sum_{\ell=2^{n-2j-2}k'}^{2^{n-2j-2}(k'+1)-1} 1 = (2^{j+1} - 1)2^{n-2j-2} \\
&= \frac{1}{2} C_d(j, k).
\end{aligned}$$

Similarly, the total number of shortest paths that go through the downward links of node  $(j, k)$  that are directed to node  $(j+1, 2k+1)$  is given by  $|\sigma(\cup_{k' \neq 2k} S(j+1, k')) \cap S(j+1, 2k+1)| = \frac{1}{2} C_d(j, k)$ . Therefore, the total number of shortest paths that go through the downward links of node  $(j, k)$  is exactly  $C_d(j, k)$ .

*Case 2:*  $\lfloor n/2 \rfloor \leq j \leq n-1$ . Clearly, the total number of shortest paths that go through the downward links of node  $(j, k)$  is bounded above by  $|S(j, k)|$ , i.e., the total number of leaves in the subtree  $T(j, k)$ . As in this case we have from (7) that  $C_d(j, k) = 2^{n-j} = |S(j, k)|$ , the proof is completed. ■

From Theorem 4, Theorem 5, and the fact that the  $N$  permutations  $P_\pi, P_{\pi_1}, P_{\pi_2}, \dots, P_{\pi_{N-1}}$ , where  $N = 2^n$ , satisfy the condition in (1), we obtain the following theorem.

**Theorem 6** *Let  $N = 2^n$ . Suppose that an  $N \times N$  fat-tree network has link capacities given by the lower bounds in (6) and (7). Then such an  $N \times N$  fat-tree network can realize the  $N$  permutations  $P_\pi, P_{\pi_1}, P_{\pi_2}, \dots, P_{\pi_{N-1}}$ , and hence can be used as the switch fabric for an  $N \times N$  load-balanced switch.*

#### IV. FULLY MESHED FAT-TREE NETWORKS

There is a very important observation from the proof of Theorem 5. Note that the total number of paths that go through the subtree  $T(j, k)$  to another subtree  $T(j, k')$  at the same level is  $|\sigma(S(j, k)) \cap S(j, k')|$  under the shortest path routing for realizing a permutation  $\sigma$  in a fat-tree network. If  $\sigma$  satisfies the uniform mapping property, then it can be seen from (18) that this number is  $2^{n-2j}$  for  $1 \leq j \leq \lfloor n/2 \rfloor$ .

In particular, for  $j = \lfloor n/2 \rfloor$ , we have  $2^{n-2j} = 1$  when  $n$  is even and we have  $2^{n-2j} = 2$  when  $n$  is odd. Therefore, the construction complexity can be greatly reduced if we simply provide direct links among the subtrees at level  $\lfloor n/2 \rfloor$  and route packets directly through these links. In other words, we replace the upper half of the tree by a simple mesh, and this leads to a much more simplified construction, called a *fully meshed fat-tree network*. Specifically, a  $2^n \times 2^n$  fully meshed fat-tree network is constructed by  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  nonblocking fat-tree networks. There are  $2^{n-2\lfloor n/2 \rfloor}$  links from each root of a  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree network to the root of another  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree network. In Figure 5, we show a  $16 \times 16$  fully-meshed fat-tree network.

To see why the  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree networks in a  $2^n \times 2^n$  fully meshed fat-tree network are nonblocking, observe that level  $j$  in the  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree networks is the level  $j + \lfloor n/2 \rfloor$  in the original  $2^n \times 2^n$  fat-tree network with link capacities given by the lower bounds in (6) and (7). Let  $\tilde{C}_u(j, k)$  (resp.,  $\tilde{C}_d(j, k)$ ) be the upward (resp., downward) capacity of node  $(j, k)$  in the  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree networks for  $1 \leq j \leq \lfloor n/2 \rfloor$  (resp.,  $0 \leq j \leq \lfloor n/2 \rfloor - 1$ ) and  $0 \leq k \leq 2^j - 1$ . Then we have from (6) and (7) that

$$\begin{aligned} \tilde{C}_u(j, k) &= C_u(j + \lfloor n/2 \rfloor, k) = 2^{n-j-\lfloor n/2 \rfloor} \\ &= 2^{\lfloor n/2 \rfloor - j}, \text{ for } 1 \leq j \leq \lfloor n/2 \rfloor, \end{aligned} \quad (19)$$

$$\begin{aligned} \tilde{C}_d(j, k) &= C_d(j + \lfloor n/2 \rfloor, k) = 2^{n-j-\lfloor n/2 \rfloor} \\ &= 2^{\lfloor n/2 \rfloor - j}, \text{ for } 0 \leq j \leq \lfloor n/2 \rfloor - 1. \end{aligned} \quad (20)$$

The link capacities in (19) and (20) are exactly the same as those required for a nonblocking  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree network.

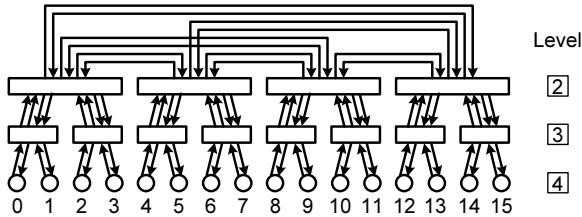


Fig. 5. A  $16 \times 16$  fully meshed fat-tree network.

By following the same argument as in the proof of Theorem 5, we also have the following theorem.

**Theorem 7** *Let  $N = 2^n$ . Consider the  $N \times N$  fully meshed fat-tree network as described in this section. Such an  $N \times N$  fully meshed fat-tree network can realize any permutation on  $Z_N$  that satisfies the uniform mapping property. In particular, it can realize the  $N$  permutations  $P_\pi, P_{\pi_1}, P_{\pi_2}, \dots, P_{\pi_{N-1}}$ , and hence can be used as the switch fabric for an  $N \times N$  load-balanced switch.*

## V. IMPLEMENTATION OF THE SWITCHES IN A FULLY MESHED FAT-TREE NETWORK

It is shown in the previous section that a  $2^n \times 2^n$  fully meshed fat-tree network is capable of realizing the needed  $2^n$

permutations for a  $2^n \times 2^n$  load-balanced switch. Although the construction complexity of a  $2^n \times 2^n$  fully meshed fat-tree network is much smaller than that of a  $2^n \times 2^n$  nonblocking fat-tree network, it still needs to implement  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  nonblocking fat-tree networks. As each node in these  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  nonblocking fat-tree networks is itself a *nonblocking* switch with many input/output ports, the construction complexity is still very high.

To further reduce the construction complexity, we will show that one does not need to implement *nonblocking* switches for the nodes in these  $2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor} \times 2^{\lfloor n/2 \rfloor}$  fat-tree networks. Specifically, for a node with upward capacity  $2^j$  and downward capacity  $2^j$ , it can be implemented by a collection of  $2^j \times 2$  switches. Our idea of achieving this is folding a specific banyan-type network.

### A. A Banyan-Type Network

In this section, we consider a  $2^n \times 2^n$  banyan-type network (see e.g., [13]), where  $n$  is an *even* number. A  $2^n \times 2^n$  banyan-type network is a multistage interconnection network with  $n$  stages, indexed from 1 to  $n$ . Each stage consists of  $2^{n-1}$   $2 \times 2$  switches, indexed from 0 to  $2^{n-1} - 1$ . As there are two inputs and two outputs in a  $2 \times 2$  switch, there are  $2^n$  inputs and  $2^n$  outputs in each stage. For each stage, index the upper input (resp., output) and the lower input (resp., output) of switch  $k$  as input (resp., output)  $2k$  and input  $2k + 1$ , respectively.

To completely specify the banyan-type network considered in this report, we need to describe how the  $2^n$  outputs from one stage are connected to the  $2^n$  inputs of the next stage. For  $0 \leq x \leq 2^n - 1$ , let  $(I_n(x), I_{n-1}(x), \dots, I_1(x))$  be the binary representation of  $x$ , where  $I_m(x)$  is the  $(n - m + 1)$ <sup>th</sup> most significant bit of  $x$  for  $1 \leq m \leq n$ . For  $1 \leq j \leq n/2 - 1$ , output  $x$  of the  $j$ <sup>th</sup> stage is connected to input  $y$  of the  $(j+1)$ <sup>th</sup> stage, where  $y$  has the following binary representation:

$$\begin{aligned} &(I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_n(x), I_{n-1}(x), \dots, I_{j+2}(x), \\ &\quad I_j(x), I_{j-1}(x), \dots, I_1(x), I_{j+1}(x)). \end{aligned} \quad (21)$$

In the middle of the banyan-type network, output  $x$  of the  $(n/2)$ <sup>th</sup> stage is connected to input  $y$  of the  $(n/2 + 1)$ <sup>th</sup> stage, where  $y$  has the following binary representation:

$$\begin{aligned} &(I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_{n/2}(x), I_{n/2-1}(x), \dots, I_1(x), \\ &\quad I_n(x), I_{n-1}(x), \dots, I_{n/2+1}(x)). \end{aligned} \quad (22)$$

Finally, for  $n/2 + 1 \leq j \leq n - 1$ , output  $x$  of the  $j$ <sup>th</sup> stage is connected to input  $y$  of the  $(j + 1)$ <sup>th</sup> stage, where  $y$  has the following binary representation:

$$\begin{aligned} &(I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_n(x), I_{n-1}(x), \dots, I_{n-j+2}(x), \\ &\quad I_1(x), I_{n-j+1}(x), I_{n-j}(x), \dots, I_2(x)). \end{aligned} \quad (23)$$

In Figure 6, we show a  $16 \times 16$  banyan-type network with such connections. Readers who are familiar with the constructions

of banyan-type networks might observe that the first  $n/2$  stages are  $2^{n/2} \times 2^{n/2} \times 2^{n/2}$  reversed baseline networks and the last  $n/2$  stages are  $2^{n/2} \times 2^{n/2} \times 2^{n/2}$  baseline networks. They are joined by a perfect shuffle in the middle. With such an observation, we will show how one can fold this banyan-type network from the middle to construct a fully meshed fat-tree network in Section V-B.

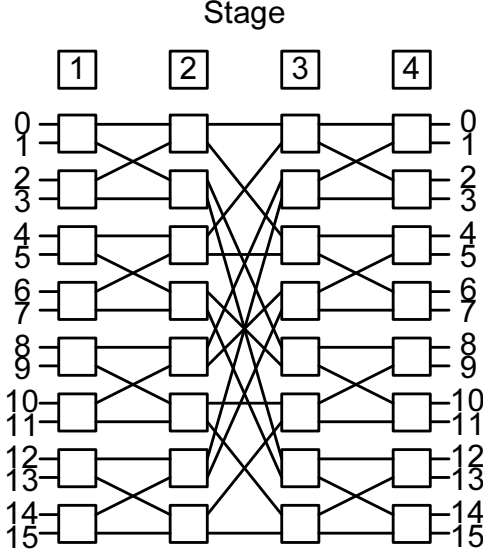


Fig. 6. A  $16 \times 16$  banyan-type network.

It is well-known that for every input/output pair in a banyan-type network, there is a unique routing path from the input to the output, and it can be used for the *self-routing* of a packet from the input to the output [13]. The unique routing path for an input/output pair is specified by setting the  $j^{\text{th}}$  switch (in the  $j^{\text{th}}$  stage) on the path according to the  $j^{\text{th}}$  most significant bit of the output. Specifically, consider an input/output pair  $(i, o)$  and let  $u_j$  (resp.,  $v_j$ ) be the input (resp., output) of the  $j^{\text{th}}$  switch on its routing path for  $1 \leq j \leq n$ . The unique routing path for the input/output pair  $(i, o)$  is specified by starting the path from input  $i$ , i.e.,  $u_1 = i$ , and setting the  $j^{\text{th}}$  switch on the path in such a way that its input  $u_j$  is connected to the upper output link (resp., lower output link) if  $I_{n-j+1}(o) = 0$  (resp.,  $I_{n-j+1}(o) = 1$ ) for  $1 \leq j \leq n$ , i.e.,

$$\begin{aligned} & (I_n(v_j), I_{n-1}(v_j), \dots, I_2(v_j), I_1(v_j)) \\ &= (I_n(u_j), I_{n-1}(u_j), \dots, I_2(u_j), I_{n-j+1}(o)). \end{aligned} \quad (24)$$

In Appendix A, we show that for  $1 \leq j \leq n/2$ , the binary representations of  $u_j$  and  $v_j$  are given as follows:

$$\begin{aligned} & (I_n(u_j), I_{n-1}(u_j), \dots, I_1(u_j)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_{j+1}(i), \\ & \quad I_n(o), I_{n-1}(o), \dots, I_{n-j+2}(o), I_j(i)) \\ & (I_n(v_j), I_{n-1}(v_j), \dots, I_1(v_j)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_{j+1}(i), \\ & \quad I_n(o), I_{n-1}(o), \dots, I_{n-j+2}(o), I_{n-j+1}(o)). \end{aligned} \quad (25)$$

Furthermore, for  $n/2 + 1 \leq j \leq n$ , the binary representations of  $u_j$  and  $v_j$  are given as follows:

$$\begin{aligned} & (I_n(u_j), I_{n-1}(u_j), \dots, I_1(u_j)) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \dots, I_{n-j+2}(o), \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{j+1}(i), I_j(i)), \\ & (I_n(v_j), I_{n-1}(v_j), \dots, I_1(v_j)) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \dots, I_{n-j+2}(o) \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{j+1}(i), I_{n-j+1}(o)). \end{aligned} \quad (26)$$

Therefore, we have from (26) (with  $j = n$ ) that  $(I_n(v_n), I_{n-1}(v_n), \dots, I_1(v_n)) = (I_n(o), I_{n-1}(o), \dots, I_1(o))$ , i.e.,  $v_n = o$ , and hence the end  $v_n$  of the unique routing path is indeed output  $o$ .

**Theorem 8** A permutation on  $Z_{2^n}$  can be realized by the  $2^n \times 2^n$  banyan-type network as described in this section if and only if the permutation has the uniform mapping property.

**Proof.** Let  $\sigma$  be a permutation on  $Z_{2^n}$ . Suppose that  $\sigma$  has the uniform mapping property. We show that  $\sigma$  can be realized by the  $2^n \times 2^n$  banyan-type network by contradiction. Assume that the routing paths for two distinct input/output pairs  $(i_1, \sigma(i_1))$  and  $(i_2, \sigma(i_2))$ , where  $i_1 \neq i_2$ , share a common link between stages  $j$  and  $j+1$  for some  $1 \leq j \leq n-1$ . It follows that the two routing paths traverse the same output of a switch in the  $j^{\text{th}}$  stage, and we have from (25) (in the case that  $1 \leq j \leq n/2$ ) and (26) (in the case that  $n/2 + 1 \leq j \leq n-1$ ) that

$$I_m(i_1) = I_m(i_2), \text{ for } j+1 \leq m \leq n, \quad (27)$$

$$I_m(\sigma(i_1)) = I_m(\sigma(i_2)), \text{ for } n-j+1 \leq m \leq n. \quad (28)$$

From (27) and (3), we see that the  $i_1, i_2 \in S(n-j, \ell)$ , where  $\ell = \sum_{m=1}^{n-j} I_{j+m}(i_1)2^{m-1}$ . From (28) and (3), we also see that  $\sigma(i_1), \sigma(i_2) \in S(j, k)$ , where  $k = \sum_{m=1}^j I_{n-j+m}(\sigma(i_1))2^{m-1}$ . It follows that  $\{\sigma(i_1), \sigma(i_2)\} \subseteq \sigma(S(n-j, \ell)) \cap S(j, k)$  and hence  $|\sigma(S(n-j, \ell)) \cap S(j, k)| \geq 2$ , contradicting to  $|\sigma(S(n-j, \ell)) \cap S(j, k)| = 1$  in (11).

Conversely, suppose that  $\sigma$  can be realized by the banyan-type network. To show that  $\sigma$  has the uniform mapping property, it suffices to show that  $|\sigma(S(n-j, \ell)) \cap S(j, k)| = 1$  for all  $0 \leq j \leq n$ ,  $0 \leq k \leq 2^j - 1$ , and  $0 \leq \ell \leq 2^{n-j} - 1$ . We first prove that  $|\sigma(S(n-j, \ell)) \cap S(j, k)| \leq 1$  for all  $0 \leq j \leq n$ ,  $0 \leq k \leq 2^j - 1$ , and  $0 \leq \ell \leq 2^{n-j} - 1$  by contradiction. Assume that  $|\sigma(S(n-j, \ell)) \cap S(j, k)| \geq 2$  for some  $0 \leq j \leq n$ ,  $0 \leq k \leq 2^j - 1$ , and  $0 \leq \ell \leq 2^{n-j} - 1$ , then there exist  $i_1 \neq i_2$  and  $i_1, i_2 \in S(n-j, \ell)$ , such that  $\sigma(i_1), \sigma(i_2) \in S(j, k)$ . It follows from (3) that (27) and (28) hold. Therefore, in the case that  $1 \leq j \leq n/2$  (resp.,  $n/2 + 1 \leq j \leq n$ ), we see from (27), (28), and (25) (resp., (27), (28), and (26)) that the routing paths for the two distinct input/output pairs  $(i_1, \sigma(i_1))$  and  $(i_2, \sigma(i_2))$  share a common link between stages  $j$  and  $j+1$ , and we have reached a contradiction.

Let  $0 \leq j \leq n$  and  $0 \leq \ell \leq 2^{n-j} - 1$ . Since  $\sigma$  is a

permutation, we have

$$\begin{aligned}
& \sum_{k=0}^{2^j-1} |\sigma(S(n-j, \ell)) \cap S(j, k)| \\
&= |\sigma(S(n-j, \ell)) \cap \cup_{k=0}^{2^j-1} S(j, k)| \\
&= |\sigma(S(n-j, \ell))| = |S(n-j, \ell)| = 2^j. \quad (29)
\end{aligned}$$

As we have already proved that  $|\sigma(S(n-j, \ell)) \cap S(j, k)| \leq 1$  for all  $0 \leq k \leq 2^j - 1$ , it is clear that (29) holds if and only if  $|\sigma(S(n-j, \ell)) \cap S(j, k)| = 1$  for all  $0 \leq k \leq 2^j - 1$ . ■

We note that a theorem similar to Theorem 8 was previously shown in Theorem 1 in [12] for the reverse-exchange network. Furthermore, the specific banyan-type network in this section can be shown to be equivalent to the baseline network and the reverse-exchange network with fixed input/output ports by using the trace and guide in Li's book [13].

### B. Folding the Banyan-Type Network

In the following, we describe how to construct a  $2^n \times 2^n$  fully meshed fat-tree network by folding the  $2^n \times 2^n$  banyan-type network specified in Section V-A (when  $n$  is an even number).

(i) For  $0 \leq i \leq 2^n - 1$ , the  $i^{\text{th}}$  input and the  $i^{\text{th}}$  output of the banyan-type network is merged as the  $i^{\text{th}}$  leaf of the fully meshed fat-tree network.

(ii) Each link from an input port to the first stage (resp., from the last stage to an output port) and each directed link from stage  $j$  to stage  $j+1$  for  $1 \leq j \leq n/2 - 1$  (resp.,  $n/2 + 1 \leq j \leq n - 1$ ) in the banyan-type network is viewed as an upward link (resp., a downward link) in the fully meshed fat-tree network.

(iii) For  $1 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ , let  $F_j(k)$  (resp.,  $F_{n-j+1}(k)$ ) be the collection of the  $2 \times 2$  switches in the  $j^{\text{th}}$  stage (resp.,  $(n-j+1)^{\text{th}}$  stage) with indices in  $S(n-j+1, k) = \{x : k \cdot 2^{j-1} \leq x \leq (k+1)2^{j-1} - 1\}$  in the banyan-type network. It is easy to see that for  $1 \leq j \leq n/2$ , the set of all of the  $2^{n-1}$   $2 \times 2$  switches in the  $j^{\text{th}}$  stage (resp., the  $(n-j+1)^{\text{th}}$  stage) is partitioned into  $2^{n-j}$  sets of switches  $F_j(0), F_j(1), \dots, F_j(2^{n-j} - 1)$  (resp.,  $F_{n-j+1}(0), F_{n-j+1}(1), \dots, F_{n-j+1}(2^{n-j} - 1)$ ), each containing  $2^{j-1}$   $2 \times 2$  switches. For  $1 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ , construct the switch at node  $(n-j, k)$  of the fully meshed fat-tree network by the collection of the  $2^j$   $2 \times 2$  switches in  $F_j(k) \cup F_{n-j+1}(k)$ .

Note that for  $1 \leq j \leq n/2 - 1$  and  $0 \leq k \leq 2^{n-j} - 1$ , each of the  $2^{j-1}$   $2 \times 2$  switches in  $F_j(k)$  has two upward links, and hence the upward capacity of the switch at node  $(n-j, k)$  is  $2^j$ , which is the same as that in (4). Also, for  $1 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ , each of the  $2^{j-1}$   $2 \times 2$  switches in  $F_{n-j+1}(k)$  has two downward links, and hence the downward capacity of the switch at node  $(n-j, k)$  is  $2^j$ , which is the same as that in (5)

Furthermore, it can be seen from the link connections in (21) for the first  $n/2$  stages in the banyan-type network that the two output links of each  $2 \times 2$  switch in  $F_j(k)$  are connected to two different  $2 \times 2$  switches in  $F_{j+1}(\lfloor k/2 \rfloor)$  for  $1 \leq j \leq n/2 - 1$

and  $0 \leq k \leq 2^{n-j} - 1$ . Therefore, the upward links of the switch at node  $(n-j, k)$  are all connected to the switch at node  $(n-j-1, \lfloor k/2 \rfloor)$  in the fully meshed fat-tree network for  $1 \leq j \leq n/2 - 1$  and  $0 \leq k \leq 2^{n-j} - 1$ . Similarly, it can be seen from the link connections in (23) for the last  $n/2$  stages in the banyan-type network that the two output links of each  $2 \times 2$  switch in  $F_{n-j+1}(k)$  are connected to one  $2 \times 2$  switch in  $F_{n-j+2}(2k)$  and another  $2 \times 2$  switch in  $F_{n-j+2}(2k+1)$  for  $2 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ . Therefore, half of the downward links of the switch at node  $(n-j, k)$  are connected to the switch at node  $(n-j+1, 2k)$  and the other half are connected to the switch at node  $(n-j+1, 2k+1)$  in the fully meshed fat-tree network for  $2 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ . Finally, the link connections in (22) for the perfect shuffle in the middle of the banyan-type network guarantee that exactly one of the  $2^{n/2}$  output links of the  $2^{n/2-1}$   $2 \times 2$  switches in  $F_{n/2}(k)$  is connected to a  $2 \times 2$  switch in  $F_{n/2+1}(k')$  for all  $0 \leq k, k' \leq 2^{n/2} - 1$ . Therefore, there is exactly one link from the switch at node  $(n/2, k)$  to the switch at node  $(n/2, k')$  in the fully-meshed fat-tree network for all  $0 \leq k, k' \leq 2^{n/2} - 1$  (note that this implies that there is an internal link inside the switch at node  $(n/2, k)$  for  $0 \leq k \leq 2^{n/2} - 1$ ). The proof of the above claims is given in Appendix B.

For example, we show in Figure 7 the folded  $16 \times 16$  banyan-type network. The  $2 \times 2$  switches represented by solid (resp., dotted) squares are from the first (resp., second) half of the banyan-type network. In particular, the switch at node  $(3, 5)$  in the fully meshed fat-tree network contains the  $2 \times 2$  switch with index 5 in the 1<sup>st</sup> stage and the  $2 \times 2$  switch with index 5 in the 4<sup>th</sup> stage of the banyan-type network.

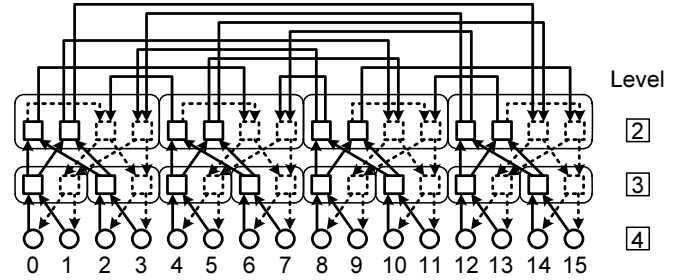


Fig. 7. Construction of a  $16 \times 16$  fully meshed fat-tree network by folding the  $16 \times 16$  banyan-type network in Section V-A.

## VI. CONCLUSION

In this report, we proposed to use *fat-tree* networks as the switch fabrics in load balanced switches. One of the main problems in fat-tree networks is that the link capacity has to be increased rapidly from the leaves to the root of the tree, and hence the complexity of implementing the switches near the root of the tree could be very high. As we only require a fat-tree network to realize a set of  $N$  permutations needed for the implementation of  $N \times N$  load-balanced switches, we showed that the implementation complexity can be greatly reduced. We first derived a lower bound on the link capacity for each switch in a fat-tree network, and then we found a

set of  $N$  permutations that achieves the lower bound by using the uniform mapping property of the bit-reversal permutation. To further reduce the implementation complexity, we also proposed a fully meshed fat-tree network that replaces the upper half of the tree by a simple mesh, and showed a fully meshed fat-tree network can be implemented by collecting  $2 \times 2$  switches in a folded banyan-type network that realizes this set of  $N$  permutations.

There are some research topics that require further study.

(i) *Incremental update of the number of linecards*: In this report, the number of input/output ports is assumed to be a power of 2. For the purpose of incremental update of the number of linecards, there are solutions by using twister networks [7] and degenerated banyan networks [8]. It seems that the approaches used there might also be applicable to our setting in this report.

(ii) *Uniform mapping property*: It is our belief that the uniform mapping property should be equivalent to the condition previously stated in Theorem 1 in [12]. As shown in Theorems 3–5 in [12], there are other sets of permutations that have the uniform mapping property. In particular, if a permutation  $\sigma$  has the uniform mapping property, then  $p\sigma$  defined by  $(p\sigma)(x) = p\sigma(x) \bmod N$  for  $x \in Z_N$ , also has the uniform mapping property when  $p$  is an odd number. This implies that one can combine the bit-reversal permutation and the  $2^n$  permutations in a  $2^n \times 2^n$  symmetric TDM switch to form another set of permutations that can also be used for implementing load-balanced switches with fat-tree networks. Further development along this direction will be reported separately.

#### APPENDIX A PROOF OF (25) AND (26)

We first note from  $u_1 = i$  and (24) (with  $j = 1$ ) that the binary representations of  $u_1$  and  $v_1$  given by

$$\begin{aligned} & (I_n(u_1), I_{n-1}(u_1), \dots, I_1(u_1)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_2(i), I_1(i)), \\ & (I_n(v_1), I_{n-1}(v_1), \dots, I_1(v_1)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_2(i), I_n(o)). \end{aligned}$$

By using (21) with  $j = 1, 2, \dots, n/2 - 1$  (in that order) and (24) with  $j = 2, 3, \dots, n/2$  (in that order), we can obtain the binary representations of  $u_j$  and  $v_j$  for  $1 \leq j \leq n/2$  as follows:

$$\begin{aligned} & (I_n(u_j), I_{n-1}(u_j), \dots, I_1(u_j)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_{j+1}(i), \\ & \quad I_n(o), I_{n-1}(o), \dots, I_{n-j+2}(o), I_j(i)) \\ & (I_n(v_j), I_{n-1}(v_j), \dots, I_1(v_j)) \\ &= (I_n(i), \dots, I_{n/2+1}(i), I_{n/2}(i), \dots, I_{j+1}(i), \\ & \quad I_n(o), I_{n-1}(o), \dots, I_{n-j+2}(o), I_{n-j+1}(o)). \end{aligned}$$

Thus, (25) is proved.

After the perfect shuffle in the middle, we have from (25) (with  $j = n/2$ ), (22), and (24) (with  $j = n/2 + 1$ ) that the binary representations of  $u_{n/2+1}$  and  $v_{n/2+1}$  are given by

$$\begin{aligned} & (I_n(u_{n/2+1}), I_{n-1}(u_{n/2+1}), \dots, I_1(u_{n/2+1})) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{n/2+2}(i), I_{n/2+1}(i)), \\ & (I_n(v_{n/2+1}), I_{n-1}(v_{n/2+1}), \dots, I_1(v_{n/2+1})) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{n/2+2}(i), I_{n/2}(o)). \end{aligned}$$

Finally, by using (23) with  $j = n/2 + 1, n/2 + 2, \dots, n - 1$  (in that order) and (24) with  $j = n/2 + 2, n/2 + 3, \dots, n$  (in that order), we can obtain the binary representations of  $u_j$  and  $v_j$  for  $n/2 + 1 \leq j \leq n$  as follows:

$$\begin{aligned} & (I_n(u_j), I_{n-1}(u_j), \dots, I_1(u_j)) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \dots, I_{n-j+2}(o), \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{j+1}(i), I_j(i)), \\ & (I_n(v_j), I_{n-1}(v_j), \dots, I_1(v_j)) \\ &= (I_n(o), I_{n-1}(o), \dots, I_{n/2+1}(o), \dots, I_{n-j+2}(o) \\ & \quad I_n(i), I_{n-1}(i), \dots, I_{j+1}(i), I_{n-j+1}(o)). \end{aligned}$$

Therefore, (26) is proved.

#### APPENDIX B PROOF OF THE CLAIMS IN SECTION V-B

In this appendix, we show the following claims in Section V-B: (i) The two output links of each  $2 \times 2$  switch in  $F_j(k)$  are connected to two different  $2 \times 2$  switches in  $F_{j+1}(\lfloor k/2 \rfloor)$  for  $1 \leq j \leq n/2 - 1$  and  $0 \leq k \leq 2^{n-j} - 1$ . (ii) The two output links of each  $2 \times 2$  switch in  $F_{n-j+1}(k)$  are connected to one  $2 \times 2$  switch in  $F_{n-j+2}(2k)$  and another  $2 \times 2$  switch in  $F_{n-j+2}(2k + 1)$  for  $2 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ . (iii) Exactly one of the  $2^{n/2}$  output links of the  $2^{n/2-1} \times 2 \times 2$  switches in  $F_{n/2}(k)$  is connected to a  $2 \times 2$  switch in  $F_{n/2+1}(k')$  for all  $0 \leq k, k' \leq 2^{n/2} - 1$ .

(i) First we show that the two output links of each  $2 \times 2$  switch in  $F_j(k)$  are connected to two different  $2 \times 2$  switches in  $F_{j+1}(\lfloor k/2 \rfloor)$  for  $1 \leq j \leq n/2 - 1$  and  $0 \leq k \leq 2^{n-j} - 1$ . To see this, let  $1 \leq j \leq n/2 - 1$  and  $0 \leq k \leq 2^{n-j} - 1$ , and consider a switch in  $F_j(k)$  with index  $x \in S(n - j + 1, k)$ . From (3) and  $I_{n-j+1}(k) = 0$  (as  $0 \leq k \leq 2^{n-j} - 1$ ), we see that the binary representation of  $x$  is given by

$$\begin{aligned} & (I_n(x), I_{n-1}(x), \dots, I_1(x)) \\ &= (0, I_{n-j}(k), I_{n-j-1}(k), \dots, I_2(k), I_1(k), \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_1(x)). \end{aligned} \tag{30}$$

Clearly, the binary representations of the two outputs  $2x$  and

$2x + 1$  of switch  $x$  in the  $j^{\text{th}}$  stage are given by

$$\begin{aligned} & (I_n(2x), I_{n-1}(2x), \dots, I_1(2x)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_2(k), I_1(k), \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_1(x), 0), \end{aligned} \quad (31)$$

$$\begin{aligned} & (I_n(2x+1), I_{n-1}(2x+1), \dots, I_1(2x+1)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_2(k), I_1(k), \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_1(x), 1). \end{aligned} \quad (32)$$

Let output  $2x$  (resp.,  $2x + 1$ ) of switch  $x$  in the  $j^{\text{th}}$  stage be connected to input  $y$  (resp.,  $z$ ) of switch  $\lfloor y/2 \rfloor$  (resp.,  $\lfloor z/2 \rfloor$ ) in the  $(j+1)^{\text{th}}$  stage. Then we have from (21), (31), and (32) that the binary representations of  $y$  and  $z$  are given by

$$\begin{aligned} & (I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_2(k), \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_1(x), 0, I_1(k)), \end{aligned} \quad (33)$$

$$\begin{aligned} & (I_n(z), I_{n-1}(z), \dots, I_1(z)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_2(k), \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_1(x), 1, I_1(k)). \end{aligned} \quad (34)$$

As we have from (33) and (34) that

$$\begin{aligned} \lfloor y/2 \rfloor &= \sum_{m=2}^{n-j} I_m(k)2^{j+m-2} + \sum_{m=1}^{j-1} I_m(x)2^m, \\ \lfloor z/2 \rfloor &= \sum_{m=2}^{n-j} I_m(k)2^{j+m-2} + \sum_{m=1}^{j-1} I_m(x)2^m + 1, \end{aligned}$$

it follows from  $\sum_{m=2}^{n-j} I_m(k)2^{j+m-2} = (k - I_1(k))2^{j-1} = \lfloor k/2 \rfloor 2^j$  and  $0 \leq \sum_{m=1}^{j-1} I_m(x)2^m \leq 2^j - 2$  that

$$\lfloor k/2 \rfloor 2^j \leq \lfloor y/2 \rfloor \leq (\lfloor k/2 \rfloor + 1)2^j - 2, \quad (35)$$

$$\lfloor k/2 \rfloor 2^j \leq \lfloor z/2 \rfloor \leq (\lfloor k/2 \rfloor + 1)2^j - 1. \quad (36)$$

As such, we see from (35), (36), and (2) that  $\lfloor y/2 \rfloor \neq \lfloor z/2 \rfloor$  and  $\lfloor y/2 \rfloor, \lfloor z/2 \rfloor \in S(n-j, \lfloor k/2 \rfloor)$ , and hence the two output links of switch  $x$  in  $F_j(k)$  are connected to two different switches in  $F_{j+1}(\lfloor k/2 \rfloor)$ .

(ii) Now we show that the two output links of each  $2 \times 2$  switch in  $F_{n-j+1}(k)$  are connected to one  $2 \times 2$  switch in  $F_{n-j+2}(2k)$  and another  $2 \times 2$  switch in  $F_{n-j+2}(2k+1)$  for  $2 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ . To see this, let  $2 \leq j \leq n/2$  and  $0 \leq k \leq 2^{n-j} - 1$ , and consider a switch in  $F_{n-j+1}(k)$  with index  $x \in S(n-j+1, k)$ . As before, the binary representations of the two outputs  $2x$  and  $2x+1$  of switch  $x$  in the  $(n-j+1)^{\text{th}}$  stage are given by (31) and (32), respectively. Let output  $2x$  (resp.,  $2x+1$ ) of switch  $x$  in the  $(n-j+1)^{\text{th}}$  stage be connected to input  $y$  (resp.,  $z$ ) of switch  $\lfloor y/2 \rfloor$  (resp.,  $\lfloor z/2 \rfloor$ ) in the  $(n-j+2)^{\text{th}}$  stage. Then we have from (23), (31), and (32) that the binary representations of  $y$

and  $z$  are given by

$$\begin{aligned} & (I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_1(k), 0, \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_2(x), I_1(x)), \end{aligned} \quad (37)$$

$$\begin{aligned} & (I_n(z), I_{n-1}(z), \dots, I_1(z)) \\ &= (I_{n-j}(k), I_{n-j-1}(k), \dots, I_1(k), 1, \\ & \quad I_{j-1}(x), I_{j-2}(x), \dots, I_2(x), I_1(x)). \end{aligned} \quad (38)$$

As we have from (37) and (38) that

$$\begin{aligned} \lfloor y/2 \rfloor &= \sum_{m=1}^{n-j} I_m(k)2^{j+m-2} + \sum_{m=2}^{j-1} I_m(x)2^{m-2}, \\ \lfloor z/2 \rfloor &= \sum_{m=1}^{n-j} I_m(k)2^{j+m-2} + 2^{j-2} + \sum_{m=2}^{j-1} I_m(x)2^{m-2}, \end{aligned}$$

it follows from  $\sum_{m=1}^{n-j} I_m(k)2^{j+m-2} = k \cdot 2^{j-1} = 2k \cdot 2^{j-2}$  and  $0 \leq \sum_{m=2}^{j-1} I_m(x)2^{m-2} \leq 2^{j-2} - 1$  that

$$2k \cdot 2^{j-2} \leq \lfloor y/2 \rfloor \leq (2k+1)2^{j-2} - 1, \quad (39)$$

$$(2k+1)2^{j-2} \leq \lfloor z/2 \rfloor \leq (2k+2)2^{j-2} - 1. \quad (40)$$

As such, we see from (39), (40), and (2) that  $\lfloor y/2 \rfloor \neq \lfloor z/2 \rfloor$ ,  $\lfloor y/2 \rfloor \in S(n-j+2, 2k)$ , and  $\lfloor z/2 \rfloor \in S(n-j+2, 2k+1)$ , and hence the two output links of switch  $x$  in  $F_{n-j+1}(k)$  are connected to one switch in  $F_{n-j+2}(2k)$  and another switch in  $F_{n-j+2}(2k+1)$ .

(iii) Finally, we show that exactly one of the  $2^{n/2}$  output links of the  $2^{n/2-1} 2 \times 2$  switches in  $F_{n/2}(k)$  is connected to a  $2 \times 2$  switch in  $F_{n/2+1}(k')$  for all  $0 \leq k, k' \leq 2^{n/2} - 1$ . To see this, let  $0 \leq k \leq 2^{n/2} - 1$ , and consider a switch in  $F_{n/2}(k)$  with index  $x \in S(n/2+1, k)$ . It can be seen that (30)–(32) still hold for  $j = n/2$ , and hence the binary representations of  $x$  and its two outputs  $2x$  and  $2x+1$  are given by

$$\begin{aligned} & (I_n(x), I_{n-1}(x), \dots, I_1(x)) \\ &= (0, I_{n/2}(k), I_{n/2-1}(k), \dots, I_1(k), \\ & \quad I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x)), \end{aligned} \quad (41)$$

$$\begin{aligned} & (I_n(2x), I_{n-1}(2x), \dots, I_1(2x)) \\ &= (I_{n/2}(k), I_{n/2-1}(k), \dots, I_1(k), \\ & \quad I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x), 0), \end{aligned} \quad (42)$$

$$\begin{aligned} & (I_n(2x+1), I_{n-1}(2x+1), \dots, I_1(2x+1)) \\ &= (I_{n/2}(k), I_{n/2-1}(k), \dots, I_1(k), \\ & \quad I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x), 1). \end{aligned} \quad (43)$$

Let output  $2x$  (resp.,  $2x+1$ ) of switch  $x$  in the  $(n/2)^{\text{th}}$  stage be connected to input  $y$  (resp.,  $z$ ) of switch  $\lfloor y/2 \rfloor$  (resp.,  $\lfloor z/2 \rfloor$ ) in the  $(n/2+1)^{\text{th}}$  stage. Then we have from (22), (42), and

(43) that the binary representations of  $y$  and  $z$  are given by

$$\begin{aligned} & (I_n(y), I_{n-1}(y), \dots, I_1(y)) \\ &= (I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x), 0, \\ & \quad I_{n/2}(k), I_{n/2-1}(k), \dots, I_2(k), I_1(k)), \end{aligned} \quad (44)$$

$$\begin{aligned} & (I_n(z), I_{n-1}(z), \dots, I_1(z)) \\ &= (I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x), 1, \\ & \quad I_{n/2}(k), I_{n/2-1}(k), \dots, I_2(k), I_1(k)). \end{aligned} \quad (45)$$

From (44) and (45), we have

$$\begin{aligned} \lfloor y/2 \rfloor &= \sum_{m=1}^{n/2-1} I_m(x)2^{n/2+m-1} + \sum_{m=2}^{n/2} I_m(k)2^{m-2}, \\ \lfloor z/2 \rfloor &= \sum_{m=1}^{n/2-1} I_m(x)2^{n/2+m-1} + 2^{n/2-1} + \sum_{m=2}^{n/2} I_m(k)2^{m-2}. \end{aligned}$$

Let  $k' = \sum_{m=1}^{n/2-1} I_m(x)2^m$  (note that  $0 \leq k' \leq 2(2^{n/2-1} - 1) = 2^{n/2} - 2$ ). It then follows from  $0 \leq \sum_{m=2}^{n/2} I_m(k)2^{m-2} \leq 2^{n/2-1} - 1$  that

$$k' \cdot 2^{n/2-1} \leq \lfloor y/2 \rfloor \leq (k' + 1)2^{n/2-1} - 1, \quad (46)$$

$$(k' + 1)2^{n/2-1} \leq \lfloor z/2 \rfloor \leq (k' + 2)2^{n/2-1} - 1. \quad (47)$$

Since  $0 \leq k', k'+1 \leq 2^{n/2} - 1$ , we see from (46), (47), and (2) that  $\lfloor y/2 \rfloor \in S(n/2 + 1, k')$  and  $\lfloor z/2 \rfloor \in S(n/2 + 1, k' + 1)$ , and hence the two output links of switch  $x$  in  $F_{n/2}(k)$  are connected to one switch in  $F_{n/2+1}(k')$  and another switch in  $F_{n/2+1}(k'+1)$ . As  $I_{n/2-1}(x), I_{n/2-2}(x), \dots, I_1(x)$  goes from  $(0, 0, \dots, 0), (0, 0, \dots, 1), \dots, (1, 1, \dots, 1)$  (note that there are  $2^{n/2-1}$   $2 \times 2$  switch in  $F_{n/2}(k)$ ), we see from  $k' = \sum_{m=1}^{n/2-1} I_m(x)2^m$  that  $(k', k' + 1)$  goes from  $(0, 1), (2, 3), \dots, (2^{n/2} - 2, 2^{n/2} - 1)$ . As such, exactly one of the  $2^{n/2}$  output links of the  $2^{n/2-1}$   $2 \times 2$  switches in  $F_{n/2}(k)$  is connected to a  $2 \times 2$  switch in  $F_{n/2+1}(k')$  for all  $0 \leq k' \leq 2^{n/2} - 1$ .

## REFERENCES

- [1] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced Birkhoff-von Neumann switches-Part I: One-stage buffering", *Computer Communications*, vol. 25, pp. 611-622, 2002.
- [2] I. Keslassy, S.-T. Chung, K. Yu, D. Miller, M. Horowitz, O. Sloggard, and N. McKeown, "Scaling Internet routers using optics," in *Proceedings ACM Special Interest Group on Data Communication (SIGCOMM'03)*, Karlsruhe, Germany, August 25-29, 2003.
- [3] I. Keslassy, S.-T. Chung, and N. McKeown, "A load-balanced switch with an arbitrary number of linecards," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'04)*, Hong Kong, China, March 7-11, 2004.
- [4] Y. Shen, S. Jiang, S. S. Panwar, and H. J. Chao, "Byte-focal: a practical load-balanced switch," in *Proceedings IEEE Workshop on High Performance Switching and Routing (HPSR'05)*, Hong Kong, China, May 12-14, 2005.
- [5] J.-J. Jaramillo, F. Milan, and R. Srikant, "Padded frames: a novel algorithm for stable scheduling in load-balanced switches," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 1212-1225, October 2008.
- [6] C.-L. Yu, C.-S. Chang, and D.-S. Lee, "CR switch: A load-balanced switch with contention and reservation", *IEEE Transactions on Communications*, vol. 17, pp. 1659-1671, October 2009.
- [7] C.-M. Lien, C.-S. Chang, J. Cheng, D.-S. Lee, and J.-T. Liao, "Twister networks and their applications to load-balanced switches," in *Proceedings IEEE International Conference on Computer Communications (INFOCOM'10)*, San Diego, CA, USA, March 15-19, 2010.

- [8] C.-M. Lien, C.-S. Chang, J. Cheng, D.-S. Lee, and J.-T. Liao, "Using banyan networks for load-balanced switches with incremental update," in *Proceedings IEEE International Conference on Communications (ICC'10)*, Cape Town, South Africa, May 23-27, 2010.
- [9] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. 34, pp. 892-901, October 1985.
- [10] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings ACM Special Interest Group on Data Communication (SIGCOMM'08)*, Seattle, WA, USA, August 17-22, 2008.
- [11] H. Hossain, M. Akbar, and M. Islam, "Extended-butterfly fat tree interconnection (EFTI) architecture for network on chip," in *Proceedings IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim'05)*, Victoria, B.C., Canada, August 24-26, 2005.
- [12] C. -L. Wu and S. -Y. Feng, "The Reverse-Exchange Interconnection Network," *IEEE Transactions on Computers*, vol. c-29, pp. 801-811, September 1980.
- [13] S.-Y. R. Li, *Algebraic Switching Theory and Broadband Applications*, San Diego, CA: Academic Press, 2001.