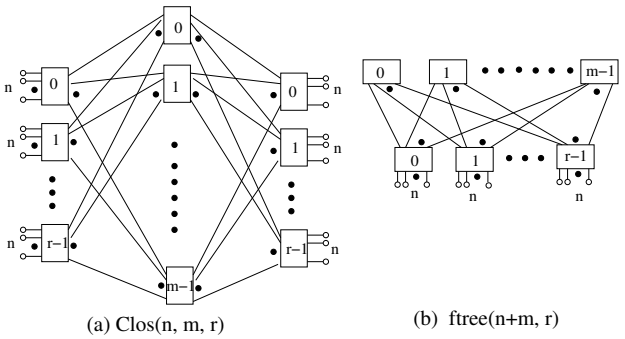
On Nonblocking Folded-Clos Networks in Computer Communication Environments

Xin Yuan
Department of Computer Science, Florida State University, Tallahassee, FL 32306
xyuan@cs.fsu.edu

***Abstract*—Folded-Clos networks, also referred to as fat-trees, have been widely used as interconnects in large scale high performance computing clusters. The switching capability of such interconnects in the computer communication environment, however, is not well understood. In particular, the concept of nonblocking interconnects, which is often used by system vendors, has only been studied in the telephone communication environment with the assumption of a centralized controller. Such "nonblocking"networks do not support nonblocking communications in computer communication environments where the network control is distributed. In this paper, we investigate folded-Clos networks that are nonblocking in computer communication environments and establish nonblocking conditions for various routing schemes including deterministic routing and adaptive routing.**



(a) Clos(n, m, r) (b) ftree(n+m, r)

Fig. 1. Clos and folded-Clos networks

I. Introduction

Clos networks and their variations such as folded-Clos (also referred to as fat-trees) have been widely used for multiprocessor interconnects and system area networks. Almost all large scale commodity high performance computing clusters are interconnected with such topologies.

A three-stage Clos network has the input stage, the middle stage, and the output stage. The input stage consists of $n \times m$ switches; the middle stage consists of $r \times r$ switches; and the output stage consists of $m \times n$ switches. There are r input switches, r output switches, and m middle switches with each of the input and output switches having a link connecting to each of the middle switches. Fig. 1 (a) depicts a three-stage Clos network. We will use the notion $Clos(n, m, r)$ to denote the Clos network with parameters n, m, and r. A folded-Clos (fat-tree) network is the one-sided version of the Clos network: it basically merges the corresponding input and output switches into one switch. Fig. 1 (b) shows a folded-Clos (fat-tree) network. $Clos(n, m, r)$ corresponds to a two level fat-tree: the lower level switches are $n + m \times n + m$ switches while the top level switches are $r \times r$ switch. We will use the notion $ftree(n + m, r)$ to denote such a fat-tree.

The switching capability of $Clos(n, m, r)$ and $ftree(n + m, r)$ is similar and is determined by the parameters n, m, and r. The study of such networks has focused on finding the most cost effective values for n, m, and r to achieve nonblocking communication, that is, the ability to establish a connection from an arbitrary input port to an arbitrary output port without causing contention. A nonblocking network can provide connectivity for any permutation communication, which consists connections from arbitrary input ports to arbitrary output ports with the restriction that each input or output port can be used at most once in the communication. The most important results in this area are summarized in various nonblocking conditions for Clos networks including *strictly nonblocking* [2], *wide-sense nonblocking* [4], [16], and *rearrangeably nonblocking* [3]. Although these nonblocking conditions results are significant, all of them were obtained with the assumption that a centralized controller is used to manage all network resources, which renders the results not applicable to computer communications where the network control is distributed.

When used in computing clusters, folded-Clos based interconnects are often treated as the replacement of central crossbar switches, which can support any permutation communication with full bisection bandwidth [7]. Although many folded-Clos based interconnects for clusters are "nonblocking" (strictly, wide sense, or rearrangeably) in theory, the delivered performance is far-from that of crossbar switches [5], [7]. Without the centralized controller that takes all connection requests and manages all network resources, even a strictly nonblocking network can still block a permutation communication.

We define *a nonblocking folded-Clos network in the computer communication environment* to be one that, with distributed control, can support any permutation communication without network contention. Using this definition, if a folded-Clos based interconnect is nonblocking, it can support any permutation communication with no contention and can thus achieve full bisection bandwidth for any permutation communication: such an interconnect behaves like a crossbar switch.

The understanding of folded-Clos based interconnects in computer communication environments is insufficient: tech-

niques for building truly nonblocking folded-Clos interconnects in computer communication environments have not been developed; it is unclear what the cost of such interconnects will be. This paper addresses these issues. We investigate folded-Clos networks that are nonblocking in computer communication environments, develop techniques for building such networks, and establish nonblocking conditions for folded-Clos networks with distributed control under various routing schemes including deterministic routing and adaptive routing. The major results include the following.

- With single-path deterministic routing, it is not cost effective to build nonblocking $ftree(n+m,r)$ with $r \leq 2n+1$. Using relatively small top level switches to build nonblocking $ftree(n+m,r)$ is not effective.
- With single-path deterministic routing, when $r \geq 2n+1$, the nonblocking condition for $ftree(n+m,r)$ is $m \geq n^2$. Using single-path deterministic routing, $O(N\sqrt{N})$-port nonblocking interconnects can be constructed using $O(N)$ N-port switches. Commonly used multipath deterministic routing schemes have the same nonblocking condition as single-path routing.
- With adaptive routing, when $r \leq n^c$, there exists a function $f(n) = O(n^{2-\frac{1}{c+1}})$ such that the nonblocking condition for $ftree(n+m,r)$ is $m \geq f(n)$. Adaptive routing allows using a smaller number of top level switches to achieve nonblocking communication in comparison to deterministic routing.

Our results advance the basic understanding of folded-Clos networks in computer communication environments. They can be used in feasibility analysis for building nonblocking folded-Clos based interconnects under other constraints, and directly applied to build such interconnects.

The rest of the paper is structured as follows. Section II discusses the related work. Section III describes the background and notations used in the paper. Section IV considers deterministic routing, derives the nonblocking condition, and presents the routing scheme for nonblocking $ftree(n+m,r)$ with single-path deterministic routing. Section V considers adaptive routing and shows that adaptive routing improves the nonblocking condition over deterministic routing. Finally, Section VI concludes the paper.

II. Related work

The switching capability of Clos networks in the context of switching telephone traffics has been extensive studied. Various nonblocking conditions have been established. A Clos network is *strictly nonblocking* if it is always possible to set up a connecting path from an idle input port to an idle output port independent of the existing connections and the path search algorithm. Clos showed [2] that $Clos(n,m,r)$ is strictly nonblocking if the number of middle stage switches $m \geq 2n-1$. A network is *wide-sense nonblocking* when it is always possible to set up a path from an idle input port to an idle output by suitably choosing routes for new connections. The conditions for a network to be wide-sense nonblocking depend on the routing algorithm; some results are obtained [4], [16]. A network is *rearrangeably nonblocking* if a new connection from an idle input port to an idle output port can always be established by rearranging the paths for existing connections. Benes [3] showed that a Clos network is rearrangeably nonblocking if $m \geq n$. All these results assume a centralized controller and cannot be directly applied to computer networks with distributed control.

After Leiserson introduced the fat-tree topology to computer networks [10], [11], the topology has become popular and extensive research has been performed on this topology. The topology has been greatly extended [13], [14]. Since traditional nonblocking networks are blocking in the computer communication environment, most studies have focused on understanding the performance of such networks by analyzing the blocking probability [6], [9], [15], or developing techniques to reduce the blocking probability [1], [6], [9], [15], [17]. Various routing techniques including randomized routing [6], [15], multi-path routing[1], [17], and adaptive routing [9], have been proposed to minimize the blocking probability. Even with all these improvements, recent studies still show that the contemporary fat-tree based interconnects offer much lower performance than crossbar switches [5], [7]. In this paper, we investigate folded-Clos networks that are truly nonblocking in computer communication environments. This research is completely different from all existing work in the area.

III. Background and notations

This paper focuses on $ftree(n+m,r)$'s that are nonblocking in computer communication environments. An example $ftree(n+m,r)$ is shown in Fig. 1 (b). There are two layers of switches and one layer of leaf nodes in the topology. The leaf nodes are communication sources and destinations. $ftree(n+m,r)$ have r bottom level $n+m$-port switches and m top level r-port switches. It supports $r \times n$ leaf nodes. The links in this topology are bidirectional. The links from leaf nodes to bottom level switches and from bottom level switches to top level switches are *uplinks*. The links from top level switches to bottom level switches and from bottom level switches to leaf nodes are *downlinks*. As shown in Fig. 1 (b), we number the m top level switches from 0 to $m-1$, the r bottom level switches from 0 to $r-1$, and the $r \times n$ leaf nodes from 0 to $r \times n - 1$. Other numbering schemes will also be used in the paper. They will be introduced before they are used.

Let us denote $(s,\ d)$ a source-destination (SD) pair with source node s to destination node d. $SRC(s,d)$ is the switch that s is in and $DST(s,d)$ is the switch d is in. We say that $SRC(s,d)$ is the source switch of (s,d) and that (s,d) starts from $SRC(s,d)$. Similarly, $DST(s,d)$ is the destination switch of (s,d) and (s,d) ends at $DST(s,d)$. A communication pattern can be represented by a set of SD pairs.

Definition 1: A *permutation communication*, or permutation, is a communication pattern where each leaf node can be the source in at most one SD pair and the destination in at most one SD pair in the communication pattern.

Property 1: Let (s_1, d_1) and (s_2, d_2) be two SD pairs in a permutation. $s_1 \neq s_2$ and $d_1 \neq d_2$.

This property can be obtained from the definition of permutation. Since $ftree(n+m, r)$ have $r \times n$ leaf nodes, a permutation communication can at most have $r \times n$ SD pairs. When all source nodes and all destination nodes are used in a permutation, there are exactly $r \times n$ SD pairs. A permutation, however, does not require all leaf nodes to be used.

To support the communication for a SD pair, a path must be used to carry the traffics for the communication. In the computer communication environment, distributed control is used. The network control is performed by the routing algorithm, which determines the routes for SD pairs. We consider several widely used routing algorithms for folded-Clos networks: single-path deterministic routing, multi-path deterministic routing, and adaptive routing. In single-path deterministic routing, one path is used to carry all traffics for each SD pair and the path for each SD pair is pre-determined. In multi-path deterministic routing, the traffics for the same SD pair are distributed among multiple pre-determined paths either in a deterministic or random manner. In adaptive routing, different paths can be used for one SD pair and the path to be used is determined dynamically based on the traffic condition.

With a given routing algorithm, when two SD pairs in a communication pattern are routed through one network link, we say that the communication pattern causes network *contention*.

Definition 2: A folded-Clos network is *nonblocking with a routing algorithm* if any permutation communication can be supported without network contention using the routing algorithm on the network.

IV. Deterministic routing

This section considers nonblocking folded-Clos networks with deterministic routing. We will first consider single-path deterministic routing and then discuss multi-path routing.

A. Single-path deterministic routing

In single-path deterministic routing, a path is deterministically assigned to each SD pair. The following lemma gives the condition for a folded-Clos network to be nonblocking with single-path deterministic routing.

Lemma 1: For any single-path deterministic routing, $ftree(n+m, r)$ is nonblocking if and only if each link carries traffics either from one source or to one destination.

Proof: We will prove the necessary condition by contradiction. Let a link L in $ftree(n+m, r)$ carries traffics from more than one source and to more than one destination. There exists at least two SD pairs, (s_1, d_1) and (s_2, d_2), $s_1 \neq s_2$ and $d_1 \neq d_2$, whose traffics are routed through L. The communication pattern that contains only these two SD pairs is a permutation. Since the routing is deterministic, there is contention on link L for this permutation and thus, the network is not nonblocking. Hence, if $ftree(n+m, r)$ is nonblocking, each link in the network carries traffics either from one source or one destination.

We will now prove the sufficient condition by contradiction. Assume that we have a permutation P that can cause network contention: there exists two SD pairs in P that are routed through one link. Let the two SD pairs be (s_1, d_1) and (s_2, d_2) and the link be L. Since P is a permutation, we have $s_1 \neq s_2$ and $d_1 \neq d_2$ (Property 1). This contradicts to the assumption that link L only carries traffics either from one source or to one destination. Hence, if each link in the network carries traffics either from one source or to one destination, $ftree(n+m, r)$ is nonblocking. □

In $ftree(n+m, r)$, each link between a leaf node and a bottom level switch only connects to one leaf node: the traffics on such a link is either to that leaf node or from the leaf node regardless of the routing algorithm. Such a link does not have contention for any permutation. The links between top level switches and bottom level switches may have contention. Since a nonblocking network must support any permutation, all possible SD pairs must be assigned a path by the routing algorithm. For SD pair (s, d), where s and d are not in the same bottom level switch, it must be routed through a top level switch and use the links between top level switches and bottom level switches. There are $r(r-1)n^2$ such SD pairs in $ftree(n+m, r)$ that must be routed carefully.

In deriving the nonblocking condition for single-path deterministic routing, we must determine the smallest m such that all of the $r(r-1)n^2$ SD pairs can be routed with each link supporting SD pairs with either the same source or the same destination. We will use a subgraph of $ftree(n+m, r)$ to analyze the number of SD pairs that can be routed through one top level switch. Fig. 2 shows the subgraph, which contains all lower level switches in $ftree(n+m, r)$, but only one top level switch. The subgraph is effectively $ftree(n+1, r)$, a regular tree topology with the root having r children and each bottom level switch having n leaf nodes.

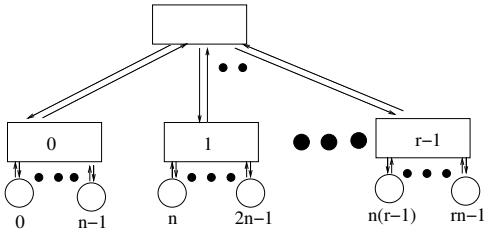


Fig. 2. The subgraph of $ftree(n+m, r)$ ($ftree(n+1, r)$)

Lemma 2: Consider using the $ftree(n+1, r)$ topology to route a subset of all possible SD pairs with source and destination in different switches. If each link can carry traffics either from one source or to one destination, then the largest number of SD pairs that can be routed through the root is at most $r \times (r-1)$ when $r \geq 2n+1$ and $2nr$ when $r \leq 2n+1$.

Proof: Let S be a largest set of SD pairs that are routed through the root switch when all links carry traffics either to one destination or from one source. To count the number of SD pairs in S, we partition the SD pairs in S into three types: (1) the SD pairs whose source switch has 2 or more sources in the

SD pairs in S, (2) the SD pairs whose destination switch has 2 or more destinations in the SD pairs in S, and (3) the SD pairs whose source switch has one source and whose destination switch has one destination in the SD pairs in S. Let us denote the number of type (1) SD pairs in S be NUM_1, the number of type (2) SD pairs be NUM_2, the number of type (3) SD pairs be NUM_3, and the total number of SD pairs in S be NUM.

Let the number of switches that have 2 or more sources in the SD pairs in S be A; the number of switches that have 2 or more destinations be B. Consider the number of type (1) SD pairs. Since there are two or more sources in each of such switches, all these sources in one switch must communicate with one destination. Otherwise, the link from the switch to the root will carries SD pairs from more than one source and to more than one destination. Hence, one switch can contribute at most n such SD pairs to NUM_1 (when all of the leaf nodes in the switch are sources). Since there are A such switches, the number of such SD pairs, NUM_1, is at most $A \times n$.

$$NUM_1 \leq A \times n.$$

Similarly,

$$NUM_2 \leq B \times n.$$

Now, consider the number of type (3) SD pairs, NUM_3. Since there are A switches that cannot be source switches for type (3) SD pairs, at most $r - A$ switches can be source switches for such SD pairs. Since each switch can at most have one destination in type (3) SD pairs, each source can at most communicate to $r - 1$ destinations (one destination in each of the $r - 1$ switches other than the source switch). Hence,

$$NUM_3 \leq (r - A) \times (r - 1).$$

Using a similar logic, by excluding switches with 2 or more destinations, we have

$$NUM_3 \leq (r - B) \times (r - 1).$$

Combine these two inequations, we obtain

$$NUM_3 \leq r \times (r - 1) - (\frac{A + B}{2})(r - 1).$$

Therefore, the total number of SD pairs,

$$\begin{aligned} NUM &\leq NUM_1 + NUM_2 + NUM_3 \\ &\leq A \times n + B \times n + r \times (r - 1) - \tfrac{A+B}{2} \times (r - 1) \\ &= r \times (r - 1) + \tfrac{A+B}{2} \times (2 \times n + 1 - r) \end{aligned}$$

When $r \geq 2n + 1$, $\frac{A+B}{2} \times (2 \times n + 1 - r) \leq 0$ and

$$NUM \leq r \times (r - 1).$$

When $r \leq 2n+1$, $\frac{A+B}{2} \times (2 \times n + 1 - r) \geq 0$. Since $A \leq r$ and $B \leq r$,

$$\begin{aligned} NUM &\leq r \times (r - 1) + \tfrac{A+B}{2} \times (2 \times n + 1 - r) \\ &\leq r \times (r - 1) + \tfrac{r+r}{2}(2 \times n + 1 - r) \\ &= 2 \times n \times r. \end{aligned}$$

□

Theorem 1: when $r \leq 2n + 1$, the number of ports supported by a nonblocking $ftree(n+m, r)$ with any single-path deterministic routing is no more than $2(n + m)$.

Proof: Regardless of the routing algorithm used, a total of $r(r-1)n^2$ SD pairs must be routed through top level switches in $ftree(n + m, r)$. From Lemma 1 and Lemma 2, when $r \leq 2n+1$, each top level switch can route at most $2 \times n \times r$ SD pairs in a nonblocking $ftree(n + m, r)$ for any single-path deterministic routing scheme. Hence, there are at least $\frac{r(r-1)n^2}{2nr} = \frac{(r-1)n}{2}$ top level switches needed; and $m \geq \frac{(r-1)n}{2}$. The number of ports supported by $ftree(n+m, r)$ is $r \times n \leq 2(\frac{r-1}{2} \times n + n) \leq 2(m + n)$. □

Theorem 1 indicates that it is not effective to build nonblocking folded-Clos networks using relatively small top level switches. When $r \leq 2n+1$, the total number of ports supported by a nonblocking folded-Clos network is at most twice that in its bottom level switches. Hence, one should focus on nonblocking folded-Clos networks with relatively large top level switches ($r \geq 2n + 1$).

Theorem 2: Let $ftree(n+m, r)$ be nonblocking with any single-path deterministic routing. When $r \geq 2n + 1$, $m \geq n^2$.

Proof: Similar to the proof of Theorem 1, regardless of the routing algorithm used, $r(r - 1)n^2$ SD pairs must be routed through top level switches. From Lemma 1 and Lemma 2, when $r \geq 2n + 1$, each top level switch can route at most $r(r - 1)$ SD pairs in a nonblocking $ftree(n + m, r)$ for any single-path deterministic routing scheme. Hence, there are at least $\frac{r(r-1)n^2}{r(r-1)} = n^2$ top level switches needed; and $m \geq n^2$. □

Theorem 2 gives the lower bound of the number of top level switches needed to make $ftree(n + m, r)$ nonblocking with any single-path deterministic routing. The following theorem establishes that this lower bound can be achieved: the $m \geq n^2$ nonblocking condition is tight.

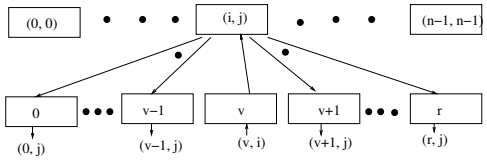
Theorem 3: There exists a single-path deterministic routing algorithm for $ftree(n + n^2, r)$ that supports all permutations without network contention. In other words, $ftree(n + n^2, r)$ is nonblocking using that routing algorithm.

Proof: We will first describe the routing algorithm and then prove $ftree(n + n^2, r)$ is nonblocking with the routing algorithm.

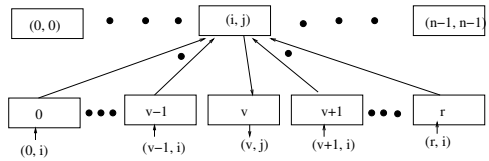
In $ftree(n+n^2, r)$, there are n^2 top level switches. We will number of n^2 top level switches by (i, j), $0 \leq i \leq n - 1$ and $0 \leq j \leq n - 1$. There are r bottom level switches numbered from 0 to $r - 1$. Each bottom level switch v, $0 \leq v \leq r - 1$, connects to n leaf nodes numbered as (v, k), $0 \leq k \leq n - 1$. The routing algorithm routes SD pair $(s = (v, i), d = (w, j))$, $0 \leq v \neq w \leq r - 1$ and $0 \leq i, j \leq n - 1$, through top level switch (i, j). That is, SD pair $(s = (v, i), d = (w, j))$ is routed through path $(v, i) \rightarrow v \rightarrow (i, j) \rightarrow w \rightarrow (w, j)$. Note that when $v = w$, $(s = (v, i), d = (v, j))$ is routed through path $(v, i) \rightarrow v \rightarrow (v, j)$.

Using this algorithm, each uplink in $ftree(n+n^2, r)$ carries traffics from one source. This obviously holds for the uplinks from leaf nodes to bottom level switches. Consider the uplink

from an arbitrary bottom level switch v to an arbitrary top level switch (i, j). There are $r-1$ SD pairs on this link: $(s = (v, i), d = (0, j))$, $(s = (v, i), d = (1, j))$, ..., $(s = (v, i), d = (v-1, j))$, $(s = (v, i), d = (v+1, j))$, ..., $(s = (v, i), d = (r, j))$. There is only one source (v, i) for all the SD pairs. Similarly, each downlink in $ftree(n + n^2, r)$ carries traffics to one destination. Consider the downlink from an arbitrary top level switch (i, j) to an arbitrary bottom level switch v. There are $r-1$ SD pairs on this link: $(s = (0, i), d = (v, j))$, $(s = (1, i), d = (v, j))$, ..., $(s = (v-1, i), d = (v, j))$, $(s = (v+1, i), d = (v, j))$, ..., $(s = (r, i), d = (v, j))$. There is only one destination (v, j) in all the SD pairs. Fig. 3 shows the SD pairs routed through the links between top level switch (i, j) and bottom level switch v. Hence, the algorithm routes SD pairs such that each link in $ftree(n + n^2, r)$ carries traffics either from at most one source or to at most one destination. It follows from Lemma 1 that the network is nonblocking. □



(a) SD pairs in the uplink from switch v to switch (i, j)



(b) SD pairs in the downlink from switch (i, j) to switch v

Fig. 3. SD pairs routed through links between top level switch (i, j) and bottom level switch v

Analysis: The main application of Clos networks is to build large (nonblocking) interconnects from smaller switches. Here, we compare nonblocking folded-Clos networks in computer communication environments with tradition rearrangeably nonblocking networks in their capability to build larger interconnects.

Consider the case when the same sized switches are used to build the nonblocking networks. That is $r = m + n$. We will use the m-port n-trees ($FT(m, n)$) [12] as a representative rearrangeably nonblocking folded-Clos in the comparison. Our two level nonblocking folded-Clos network, $ftree(n+n^2, n+n^2)$, uses $2n^2 + n$ $n^2 + n$-port switches to support $n^3 + n^2$ nonblocking ports. Let $N = n^2 + n$, our nonblocking network uses roughly $2N$ N-port switches to support roughly $N^{\frac{3}{2}}$ nonblocking ports. Traditional $FT(N, 2)$ uses $\frac{3N}{2}$ N-port switches to support $\frac{N^2}{2}$ ports [12]. Table I compares that number of ports and the number of switches needed for the two types of folded-Clos networks.

We emphasize that our nonblocking networks behave like crossbar switches while $FT(m, n)$ is not nonblocking in the computer communication environment. It is thus expected that our nonblocking networks are more expensive to construct.

building block size	$ftree(n + n^2, n + n^2)$		$FT(n + n^2, 2)$	
	# of switches	# of ports	# of switches	# of ports
20-port $(4 + 4^2)$	36	80	30	200
30-port $(5 + 5^2)$	55	150	45	450
42-port $(6 + 6^2)$	88	252	63	884

TABLE I
SIZE OF NONBLOCKING $ftree(n + n^2, n + n^2)$ AND $FT(n + n^2, 2)$

Our technique allows larger nonblocking folded-Clos networks to be constructed from smaller switches as shown in Table I, and is optimal for building such networks with single-path deterministic routing.

To support larger numbers of ports, the method to build 2-level nonblocking folded-Clos networks can be recursively applied to build more levels of nonblocking folded-Clos networks. For example, to obtain a 3-level nonblocking network, a 2-level nonblocking network can be used to replace each of either the top level switches or the bottom level switches in 2-level networks. Since our nonblocking $ftree(n + m, r)$ supports all permutations with no contention, it can be shown by induction that the recursively-built larger network will also support all permutations with no contention and is thus nonblocking. One question is whether it is more effective to replace bottom level switches or top level switches. Theorem 1 gives the answer to this question: it is more beneficial to have large top level switches. Hence, when building more than two levels folded-Clos networks, one should replace top level switches with nonblocking networks. Using this approach, a three-level nonblocking folded-Clos network built with $n+n^2$-port switches will resemble $ftree(n+n^2, n^3+n^2)$, with each top level $n^3 + n^2$-port switch being realized with a $ftree(n + n^2, n + n^2)$. This nonblocking network has $2n^4+3n^3+n^2$ n^2+n-port switches and supports n^4+n^3 ports. Let $N = n + n^2$. The three-level nonblocking folded-Clos network uses $O(N^2)$ $O(N)$-port switches to obtain an $O(N^2)$-port nonblocking network. As a comparison, $FT(N, 3)$ uses $O(N^2)$ $O(N)$-port switches to obtain $O(N^3)$-port fat-trees.

B. Multipath routing

Besides single-path deterministic routing, other routing schemes such as multipath routing [12] have also been developed for folded-Clos networks. For example, InfiniBand allows multiple paths to be set-up between two end-points [8]. In multipath routing, the traffics for the same SD pair are distributed among multiple paths either in a deterministic or random manner. Single-path routing is a form of multipath routing. Multipath routing schemes usually can achieve better load balance in folded-Clos networks than single-path routing [17]. For nonblocking folded-Clos networks, however, splitting traffics from one SD pairs to multiple paths does not improve the nonblocking condition: assuming that the timing for using different paths for a SD pair is unpredictable, which is common for multipath routing, to achieve nonblocking

communication for any permutation, Lemma 1 still needs to hold, which leads to the same bound in Theorem 2 ($m \geq n^2$) for multipath deterministic routing.

V. Adaptive routing

This section studies nonblocking $ftree(m+n, r)$ with adaptive routing. For $ftree(n+m, r)$, the adaptivity is achieved only in input switches (bottom level). Once a packet reaches the top level switch, there is only one path to each destination and no adaptivity is possible. Intuitively, adaptive routing allows more flexibility than deterministic routing and thus, may require a smaller number of top level switches to achieve nonblocking communication for any permutation. We will show in this section that adaptive routing indeed improves the nonblocking condition over deterministic routing, which indicates that more cost effective nonblocking $ftree(n+m, r)$ can be built with adaptive routing than with deterministic routing.

Since the adaptivity is achieved by input switches, for a given permutation P, we can partition the SD pairs in P into r disjoint subsets P^0, P^1, ..., P^{r-1} where P^i, $0 \leq i \leq r-1$, contains all SD pairs whose sources are in bottom level switch i. We will use the phrase *SD pairs from the same switch* to denote SD pairs whose sources are in the same switch, *SD pairs from different switches* to denote SD pairs whose sources are in different switches, *SD pairs to the same switch* to denote SD pairs whose destinations are in the same switch, and *SD pairs to different switches* to denote SD pairs whose destinations are in different switches. Our routing algorithm has the following assumptions.

- For a given set of SD pairs whose the sources are in one bottom-level switch, the switch can use any possible route for each of the SD pairs. This assumption models adaptive routing.
- There is no global information shared among different switches. Each input switch adapts based on its local traffic pattern. Routing and traffic patterns in other switches do not affect the routing decision. Note that routing based on global information has the same effect as having a centralized controller.

Lemma 3: Let (s_1, d_1) and (s_2, d_2) be two arbitrary SD pairs where $d_1 \neq d_2$ are in the same switch. If an adaptive routing algorithm for $ftree(n+m, r)$ guarantees to route such two SD pairs through different top level switches, then routes for any two SD pairs in any permutation whose sources are in different input switches will not have network contention using the adaptive routing algorithm.

Proof: Let us denote $SRC(s, d)$ the source switch of SD pair (s, d) and $DST(s, d)$ the destination switch of SD pair (s, d). Let (s_1, d_1) and (s_2, d_2) to be arbitrary two SD pairs in a permutation where s_1 and s_2 are in different switches ($SRC(s_1, d_1) \neq SRC(s_2, d_2)$). We will prove that under the assumption in the lemma, these two SD pairs will not have contention. Let (s_1, d_1) be routed through top level switch A and (s_2, d_2) be routed through top level switch B (A and B may be the same). Since $SRC(s_1, d_1) \neq SRC(s_2, d_2)$, uplinks $SRC(s_1, d_1) \rightarrow A$ and $SRC(s_2, d_2) \rightarrow B$ are different regardless whether $A = B$ or not, and there is no contention in the uplinks for the two SD pairs. For the downlink $A \rightarrow DST(s_1, d_1)$ and $B \rightarrow DST(s_2, d_2)$, there are two cases. When $DST(s_1, d_1) = DST(s_2, d_2)$, since (s_1, d_1) and (s_2, d_2) are from one permutation communication, $d_1 \neq d_2$. By the assumption of this lemma, we have $A \neq B$ and thus, the downlinks for the two SD pairs are different and there is no network contention. When $DST(s_1, d_1) \neq DST(s_2, d_2)$, regardless whether $A = B$ or not, the downlinks $A \rightarrow DST(s_1, d_1)$ and $B \rightarrow DST(s_2, d_2)$ are different and there is no contention. □

This lemma indicates that using a class of adaptive routing algorithms that guarantee to use different top level switches to route SD pairs with different destinations in the same switch, SD pairs from different switches in a permutation will not have contention. We will use the term **class DIFF** to denote this class of routing algorithms. This has two implications. First, using a class DIFF algorithm, SD pairs from different switches can be routed independently. Second, to achieve nonblocking communication, a class DIFF algorithm can focus on avoid contention for SD pairs from one switch: SD pairs from different switches will not have contention. Class DIFF algorithms are the base for our proposed adaptive routing algorithm for nonblocking $ftree(n+m, r)$.

For $ftree(n+m, r)$, there exists a c such that $r \leq n^c$. For practical folded-Clos networks, c is a small constant. For example, in $ftree(n+m, n^2)$, $c = 2$. In $ftree(n+m, n^2+n)$, $c = 3$. In our adaptive routing algorithm, we will number the r bottom level switches with c n-based digits: $s_{c-1}s_{c-2}...s_0$, $0 \leq s_i \leq n-1$ for all $0 \leq i \leq c-1$; and we will number the $r \times n$ leaf nodes in $ftree(n+m, r)$ with $c+1$ n-based digits: $s_{c-1}s_{c-2}...s_0p$, where $s_{c-1}s_{c-2}...s_0$ is the switch that the leaf node is in, and $0 \leq p \leq n-1$ is the local node number within switch $s_{c-1}s_{c-2}...s_0$.

Our adaptive routing algorithm for $ftree(n+m, r)$ schedules the SD pairs from each switch separately. SD pairs from each switch are routed in phases. In each phase, SD pairs are routed over $(c+1) \times n$ top level switches. We will use the term **configuration** to denote the group of $(c+1) \times n$ top level switches used in one scheduling phase. The routing algorithm is the same for different configurations. Within each configuration, we further partition the $(c+1) \times n$ top level switches into $c+1$ groups of n switches. We will call each of the groups of n top level switches, a *partition*. Each of the $c+1$ partitions is to schedule different types of SD pairs using a class DIFF routing scheme. In each of the partitions, the n top level switches are numbered from 0 to $n-1$.

The first partition is used to route traffics to destinations with different local node numbers. In this partition, top level switch i is only used to carry SD pairs with destinations whose local node number $p = i$ for all bottom level switches: the routing algorithm makes sure that only SD pairs whose destinations have different local node numbers are routed through this partition.

Lemma 4: Let $(s_1, d_1 = s^1_{c-1}...s^1_0p^1))$, $(s_2, d_2 =$

$s^2_{c-1}...s^2_0p^2)$), ..., and $(s_k, d_k = s^k_{c-1}...s^k_0p^k))$ be a set of SD pairs from the same (arbitrary) switch in a permutation, where $p^1, p^2, ..., p^k$ are different numbers. All of the k SD pairs can be routed through the first partition without contention.

Proof: We will first prove that the routes for $(s_1, d_1 = s^1_{c-1}...s^1_0p^1))$, $(s_2, d_2 = s^2_{c-1}...s^2_0p^2))$, ..., and $(s_k, d_k = s^k_{c-1}...s^k_0p^k))$, do not have contention with the routes for SD pairs from other switches, and then show that routes for the SD pairs do not have contention between each other. Using the routing scheme, SD pairs whose destinations have a local node number p, $0 \leq p \leq n-1$, are routed through top level switch p: the routing algorithm is a class DIFF algorithm. From Lemma 3, the routes for $(s_1, d_1 = s^1_{c-1}...s^1_0p^1))$, $(s_2, d_2 = s^2_{c-1}...s^2_0p^2))$, ..., and $(s_k, d_k = s^k_{c-1}...s^k_0p^k))$ will not have contention with the routes for SD pairs from other switches.

The routing algorithm routes $(s_i, d_i = s^i_{c-1}...s^i_0p^i)$, $0 \leq i \leq k$, to top level switch p^i. Since $p^1, p^2, ..., p^k$ are different: the paths for all of the SD pairs are link-disjoint and do not have contention. Thus, all of the k SD pairs can be routed through the first partition without contention. □

Lemma 4 implies that at least one SD pair in any permutation from each switch can be routed through this partition without contention. Since each switch can have at most n SD pairs in a permutation, at most n such partitions are needed to achieve nonblocking communication for any permutation. However, n such partitions have n^2 top level switches. Thus, this naive scheme does not improve nonblocking condition in comparison to single-path deterministic routing.

The first partition is used to route SD pairs to destinations with different local node numbers, which is the first digit in the $c+1$ n-based digits numbering scheme for leaf nodes. The second partition is used to route SD pairs to destinations with different second digit values using a class DIFF routing scheme. In the second partition, top level switch i carries traffics (from any sources) to destinations $(s_{c-1}...s_1(s_0 = i)(p = 0))$, $(s_{c-1}...s_1(s_0 = (i+1)\%n)(p = 1))$,..., $((s_{c-1}...s_1(s_0 = (i+j)\%n)(p = j))$, ..., $((s_{c-1}...s_1(s_0 = (i+n-1)\%n)(p = n-1))$. Here, % is the module operation. Using this routing scheme, different destinations in the same switch are also routed through different top level switches. For an arbitrary bottom level switch $s_{c-1}...s_0$, SD pairs with destination $s_{c-1}...s_00$ are routed through top level switch switch s_0, destination $s_{c-1}...s_01$ through top level switch $(s_0-1)\%n$, ..., and destination $s_{c-1}...s_0j$, $0 \leq j \leq n-1$, through top level switch $(s_0-j)\%n$ in the partition. Clearly, SD pairs with different destinations in the same switch are routed through different top level switches. Thus, the routing for the second partition is also a class DIFF routing algorithm.

The routing in other $c-1$ partitions is similar to that in the second partition. Basically, the i-th partition, $2 \leq i \leq c+1$, is used to route SD pairs to destinations with different i-th digit values (different s_{i-2} values) using a class DIFF routing algorithm. Specifically, in the i-th partition, top level switch i carries traffics to destinations $(s_{c-1}...(s_{i-2} = i)...s_0(p = 0))$, $(s_{c-1}...(s_{i-2} = (i+1)\%n)...s_0(p = 1))$, ..., $((s_{c-1}...(s_{i-2} = (i+j)\%n))...s_0(p = j))$, ..., $((s_{c-1}...(s_{i-2} = (i+n-1)\%n)...s_0(p = n-1))$. Following a similar logic for the routing in the second partition, the routing for the i-th partition, $2 \leq i \leq c+1$, is a class DIFF routing algorithm.

Lemma 5: Let $(s_1, d_1 = s^1_{c-1}...s^1_0p^1)$, $(s_2, d_2 = s^2_{c-1}...s^2_0p^2)$, ..., $(s_k, d_k = s^k_{c-1}...s^k_0p^k)$ be a set of SD pairs from the same switch in a permutation, where for one i, $2 \leq i \leq c+1$, s^1_{i-2}, s^2_{i-2}, ..., s^k_{i-2} are different. All of the k SD pairs can be routed through the i-th partition without contention

Proof: Since the routing in the i-th partition is a class DIFF routing algorithm, from Lemma 3, routes for the set of SD pairs will have no contention with routes for SD pairs from other switches in the i-th partition.

In the i-th partition, $(s_j, d_j = s^j_{c-1}...s^j_0p^j)$, $1 \leq j \leq k$, is routed to top level switch s^j_{i-2}. Since s^1_{i-2}, s^2_{i-2}, ..., s^k_{i-2} are different: the paths for all the SD pairs are link-disjoint. Thus, All of the k SD pairs can be routed through the i-th partition without contention. □

Lemma 5 implies that at least one SD pair from each switch can be routed through the i-th partition, $2 \leq i \leq c+1$. Lemma 4 and Lemma 5 show that each of the partitions in a configuration can be used to route a set of SD pairs from each switch in a permutation whose destinations differ in one digit in the $c+1$ digits n-based representation. We will call a set of SD pairs that can be routed through a partition without causing contention *the set of SD pairs that can be routed through the partition*. Moreover, the routing schemes in all partitions are class DIFF schemes, which guarantees that SD pairs from different switches routed through the same partition do not have contention.

Our adaptive routing algorithm for nonblocking $ftree(n+m, r)$, called NONBLOCKINGADAPTIVE, is presented in Fig. 4. Since the routing in each partition is a class DIFF routing algorithm, routing SD pairs from different switches for the same partition will not have contention and can be done independently. The routing algorithm routes SD pairs from each source switch independently. It considers configurations one at a time, greedily finds the largest number of SD pairs that can be routed through one of the unused partitions in each configuration, and routes the SD pairs to partition. This scheme is repeated with more configurations until all SD pairs are routed. After the configurations for SD pairs from all switches are computed, the algorithm merges the routes for SD pairs from different source switches (lines (14) and (15)): the corresponding partitions in each configuration for SD pairs from different switches can be routed through the same n top level switches without contention. In the description, we assume that there are sufficient number of top level switches to be allocated. We will prove the upper bound of the number of top level switches required for this algorithm in Theorem 5.

Theorem 4: Algorithm NONBLOCKINGADAPTIVE results in nonblocking communication.

Proof: The algorithm routes SD pairs in Lines (7) and (8) in Fig 4. Since it always routes SD pairs that can be routed on a partition to the partition, from lemma 4 and Lemma 5, the

Algorithm NONBLOCKINADAPTIVE:
Input: A permutation P
Output: the routes for all SD pairs in P

(1) Let P^i, $0 \leq i \leq r-1$, be the set of SD pairs in P from switch i;
(2) For each P^i, $0 \leq i \leq r-1$ do
(3) $x_i = 0$;
(4) While (P^i is not empty) do
(5) $C^i_{x_i}$ = new configuration; x_i++;
(6) While ((P^i is not empty) and (C^i_x has unused partitions)) do
(7) Find the largest subset of P^i, $LSET$, that can be routed on one of the unused partition, $PART$, in $C^i_{x_i}$;
(8) Route SD pairs in $LSET$ on $PART$;
(9) Mark $PART$ as used;
(10) $P^i = P^i - LSET$;
(11) End while
(12) End while
(13) End for
(14) Let $totalconf$ be the largest x_i, $0 \leq i \leq r-1$;
(15) For j=0 to $totalconf - 1$ do merge C^i_j, $0 \leq i \leq r-1$, into one configuration;
(16) End for

Fig. 4. An adaptive routing algorithm for nonblocking $ftree(n+m, r)$

communication is nonblocking for any permutation. □

Analyzing the number of top level switches needed by the algorithm is more challenging. Before we prove the upper bound for the number of top level switches needed in NONBLOCKINGADAPTIVE, we will show that this algorithm requires less than n^2 top level switches for any permutation in $ftree(n+m, r)$. For any two SD pairs from a switch in a permutation, the destinations are different: they differ in at least one digit in the $c+1$ n-based digits representation. The two SD pairs can be routed through one partition in a configuration. Hence, the $LSET$ found in line (7) in the first iteration after a new configuration is allocated will have at least two SD pairs when there are more than one SD pair in P^i. In other iterations, when P^i is not empty, the $LSET$ at least contains one SD pair from each switch since each unused partition can at least route one SD pair for each switch with no contention. Hence, the $c+1$ partitions in one configuration can route at least route $c+2$ SD pairs for a source switch. Since each switch can have at most n SD pairs in a permutation, at most $\frac{n}{c+2}$ configurations and $\frac{n}{c+2} \times cn = \frac{c+1}{c+2}n^2$ top level switches are needed. In the following, we will show that NONBLOCKINGADAPTIVE improve the nonblocking condition asymptotically.

Lemme 6: Consider a set of numbers encoded with $c+1$ n-based digits, $d_c d_{c-1} ... d_0$. For any k different numbers, there exists at least one i, $0 \leq i \leq c$, such that there are at least $k^{\frac{1}{c+1}}$ numbers in the set of k different numbers with different d_i.

Proof: Let the number of different values of each digit d_i among the k different numbers be X_i. If $X_i < k^{\frac{1}{c+1}}$ for all $0 \leq i \leq c$, then at most $X_0 \times X_1 \times ... \times X_c$ different numbers can be in the set. Hence, $k \leq X_0 \times X_1 \times ... \times X_c < k$, which cannot be true. Hence, there exists at least one i such that $X_i \geq k^{\frac{1}{c+1}}$. □

Theorem 5: Let $r \leq n^c$, where c is a constant. Algorithm NONBLOCKINGADAPTIVE requires at most $O(n^{2-\frac{1}{c+1}})$ top level switches to route any permutation in $ftree(n+m, r)$.

Proof: Consider the number of configurations needed by NONBLOCKINGADAPTIVE for any permutation communication. Each source switch will have at most n SD pairs to route in a permutation. From Lemma 6, among the n different destinations in the n SD pairs, which are represented by $c+1$ n-based digits, there exists an i, $0 \leq i \leq c$, such that there are $n^{\frac{1}{c+1}}$ destinations whose encoding differs in the i-th digit. A set of $n^{\frac{1}{c+1}}$ SD pairs with one for each of those destinations can be routed through a partition (Lemma 4 and Lemma 5). Since the largest subset of P^i that can be routed on one partition is found in line (7) in Fig. 4, the first partition after each new configuration is allocated will route at least $n^{\frac{1}{c+1}}$ SD pairs for each source switch when $|P^i| = n$. Let $T(n)$ be the number of configurations needed when each switch has n SD pairs in a permutation to be routed. We have $T(n) \leq T(n - n^{\frac{1}{c+1}}) + 1$.

For $n > X > \frac{n}{2}$, $n^{\frac{1}{c+1}} > X^{\frac{1}{c+1}} > (\frac{n}{2})^{\frac{1}{c+1}}$. Hence,

$$\begin{aligned} T(n) &\leq T(n/2) + \frac{\frac{n}{2}}{(\frac{n}{2})^{\frac{1}{c+1}}} \\ &= T(n/2) + (\frac{n}{2})^{1-\frac{1}{c+1}} \\ &\leq (\frac{n}{2})^{1-\frac{1}{c+1}}(1 + (\frac{1}{2})^{1-\frac{1}{c+1}} + (\frac{1}{2})^{2\times(1-\frac{1}{c+1})} + ...) \\ &= O(n^{1-\frac{1}{c+1}}) \end{aligned}$$

Hence, the number of top level switches needed for the algorithm is not more than $T(n) \times c \times n = O(n^{2-\frac{1}{c+1}})$, assuming c is a constant. □

Algorithm NONBLOCKINGADAPTIVE asymptotically improves the nonblocking condition for $ftree(n+m, r)$ with deterministic routing. However, it may not achieve the lower bound of m for adaptive routing. The tight nonblocking condition for $ftree(n+m, r)$ with adaptive routing is still open for investigation.

VI. Conclusion

We study folded-Clos networks that are nonblocking in computer communication environments and develop techniques to construct such networks. We show that it is not effective to build nonblocking folded-Clos with small top level switches. We prove that for $ftree(n+m, r)$ with single-path deterministic routing, the nonblocking condition is $m \geq n^2$. We give the single-path routing scheme that can be used to build nonblocking $ftree(n+n^2, r)$, which is optimal. We further prove that using adaptive routing, the nonblocking condition can be improved over deterministic routing. Our results indicate that it is possible to build large nonblocking

folded-Clos networks in computer communication environments using smaller components. This paper leaves an open problem to be investigated in the future: what is the tight nonblocking condition for $ftree(n + m, r)$ with adaptive routing?

ACKNOWLEDGMENT

The author thanks Prof. Piyush Kumar for providing the proof for the bound derived from the recurrence $T(n) \leq T(n - n^{\frac{1}{c+1}}) + 1$, which is used in the proof of Theorem 5.

REFERENCES

[1] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali, "Adaptive Source Routing in Multistage Interconnection Networks," the *10th International Parallel Processing Symposium*, pages 258-267, 1996.

[2] C. Clos, "A Study of Non-Blocking Switching Networks," *Bell System Technical Journal*, 32:406-424, 1953.

[3] V.E. Benes, "On Rearrangeable Three-Stage Connecting Networks," *Bell System Technical Journal*, 41(5):1481-1492, Sept. 1962.

[4] V.E. Benes, "Mathematical Theory of Connecting Networks and Telephone Traffic," Tcademic Press, 1965.

[5] P. Geoffray and t. Hoefler, "Adaptive Routing Strategies for Modern High Performance Networks," *the 16th IEEE Symposium on High Performance Interconnects*, pages 165-172, August 2008.

[6] R. I. Greenberg and C. E. Lerserson, "Ramdonzied Routing on Fat-trees." In *26th Annual IEEE Symposium on Foundations of Computer Science*, pages 241-249, Oct. 1985.

[7] T. Hoefler, T. Schneider, and A. Lumsdain, "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks," *IEEE International Conference on Cluster Computing*, pages 116-125, 2008.

[8] Infiniband^{TM} Trade Association, *Infiniband ^{TM} Architecture Specification*, Release 1.2, October 2004.

[9] J. Kim, W. J. Dally, and D. Abts, "Adaptive Routing in High-Radix Clos Network," *ACM SC'2006*, November 2006.

[10] C.E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transactions on Computers*, 34(10):892-901, October 1985.

[11] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong-Chan, S-W. Yang, and R. Zak, "The network architecture of the Connection Machine CM-5." *Journal of Parallel and Distributed Computing*, 33(2):145–158, Mar 1996.

[12] X. Lin, Y. Chung, and T. Huang, "A Multiple LID Routing Scheme for Fat-Tree-Based Infiniband Networks." *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium* (IPDPS'04), p. 11a, Sana Fe, NM, April 2004.

[13] S.R. Ohring, M. Ibel, S.K. Das, M.J. Kumar, "On Generalized Fat Trees," *International Parallel Processing Symposium*, pages 37-44, April 1995.

[14] F. Petrini and M. Vanneschi, "k-ary n-trees: High Performance Networks for Massively Parallel Architectures," *International Parallel Processing Symposium* (IPPS), pages 87-87, 1997.

[15] A. Singh, "Load-Balanced Routing in Interconnection Networks," PhD thesis, Stanford University, 2005.

[16] Y. Yang and J. Wang, "Wide-Sense Nonblocking Clos Networks under Packing Strategy," *IEEE Trans. on Computers*, 48(3):265-284, March 1999.

[17] X. Yuan, W. Nienaber, Z. Duan, and R. Melhem, "Oblivious Routing in Fat-tree Based System Area Networks with Uncertain Traffic Demands," *ACM SIGMETRICS*, pages 337-348, June 2007.