A Class of Interconnection Networks for Multicasting

Yuanyuan Yang, *Member*, *IEEE*

Abstract—Multicast, or one-to-many, communication arises frequently in parallel computing and telecommunication applications. Multicast networks can simultaneously support multiple multicast connections between the network inputs and network outputs. However, due to the complex communication patterns and routing control in multicast networks, there is still a considerably large gap in network cost between the currently best known multicast networks and permutation networks. In this paper, we will present a class of interconnection networks which can support a substantial amount of well-defined multicast patterns in a nonblocking fashion and yet have a comparable cost to permutation networks. We will also provide an efficient routing algorithm for satisfying multicast connection requests in such networks. Moreover, the multicast capability of the networks will be represented as a function of fundamental network structural parameters so that the trade-off between the network multicast capability and the network cost can be determined.

Index Terms—Interconnection networks, collective communication, multicast, nonblocking, routing algorithms.

1 Introduction

Multicast, or *one-to-many*, communication is a fundamental *collective communication* operation [1] and is highly demanded in parallel computing and telecommunication applications. Examples of such applications include *Fast Fourier Transform* (FFT), barrier synchronization [1], and write update/invalidate in directory-based cache coherence protocols. Also, teleconferencing and video broadcasting are typical applications in a telecommunication environment.

There has been growing interest in supporting multicast in parallel computers. Multicast can be supported in either hardware or software. For example, the nCUBE-2 supports broadcast and a form of restricted multicast in hardware, but, since its interconnection network is a *direct network* in which each node has a dedicated link to each of its neighbor nodes, the routing algorithm adopted may cause a deadlock when two packets are sent at the same time. This is because there may be cyclic dependency between channels in a direct network [2]. This situation becomes worse in multicast communication. There has also been much work on supporting multicast in wormhole routed direct networks in software [2]. The basic approach is sending a message along a subset of nodes on a "multicast tree." This approach needs at least $\log N$ steps to send a message to N destinations. On the other hand, among the parallel computers using *indirect networks* or *multistage networks*, both IBM GF11 and NEC Cenju-3 support a form of restricted multicast in hardware. In IBM GF11, a multicast may need two passes through the network and, in NEC Cenju-3, only single multicast is supported. In addition, there has been some work on supporting multicast in software in multistage networks [2].

Since multistage networks can have deadlock-free routing and equal communication latency between any network inputs and outputs (when packet recirculation is not allowed) [2], they receive more and more attention for the interconnecting needs of parallel computers [3] and ATM switch architectures in broadband

• *The author is with the Department of Computer Science, University of Vermont, Burlington, VT 05405. E-mail: yang@cs.uvm.edu.*

Manuscript received 9 May 1996; revised 15 Nov. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 106496.

networks. Meanwhile, since multicast is a fundamental communication pattern in many parallel applications, fast implementation for it will significantly reduce the execution time of such applications. Thus, supporting multicast in parallel computers has become an increasingly important issue [4], [5]. In this paper, we will be concerned with providing cost-effective hardware support for multiple multicasts in multistage networks.

A *multicast connection* in a multistage network can connect a network input port simultaneously to more than one network output port. In the following, we refer to a maximal set of multicast connections between the inputs and outputs of a multistage network as a *multicast assignment*. Then, a *multicast network* is a network which can realize all possible multicast assignments.

Multicast networks have been extensively studied and much progress has been made in this area [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. However, the perceived high network cost and complex routing control of multicast networks might still discourage system designers from seriously considering them for practical parallel computing systems and other communication systems. In fact, due to the complex communication patterns in multicast networks, there is still a considerably large gap in network cost between the currently best known multicast networks and permutation networks. Meanwhile, many real applications may not need full multicast capability. Although permutation networks with multicast switches may realize some multicast patterns, they in general cannot satisfy the needs of such applications. This is because permutation networks are designed for realizing only one-to-one connections and there may not be a clear definition of the type of multicast patterns a permutation network can realize. This drawback of permutation networks may prevent software and algorithm designers of parallel computing systems from efficiently utilizing multicast capability since there is not a simple rule for them to judge whether a given multicast connection can be routed in a single pass through the network.

As discussed above, full multicast networks are still too expensive for practical multicast applications and permutation networks in general cannot support multicast efficiently. Hence, we are motivated to consider a compromising network design for practical multicast applications. In this paper, we will propose a class of interconnection networks which can realize a substantial amount of well-defined multicasts and yet have a comparable cost to permutation networks. We will refer to such networks as *restricted multicast networks*. We will also provide an efficient routing algorithm for satisfying multicast connection requests in such networks.

The rest of this paper is organized as follows: Section 2 gives the necessary definitions and notations for restricted multicast networks. Section 3 reviews the previous results related to this type of networks for both permutation and multicast. Section 4 presents the main results of the paper, the nonblocking conditions for the proposed restricted multicast networks. The routing algorithm is described in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we present some basic definitions and notations that will be useful in our analysis of restricted multicast networks.

The network to be considered is a class of networks based on the Clos network [6]. This type of network belongs to so-called *constant stage networks* or *limited stage networks*. Since the network latency of a network is proportional to the number of stages in the network, a constant stage network can guarantee a short constant latency, regardless of the number of processor or memory modules in a parallel computing system, whereas most of other networks (i.e., so-called *growing stage networks*) [15], [16], [17], [18], [19] require at least log N stages for an $N \times N$ network, which represents

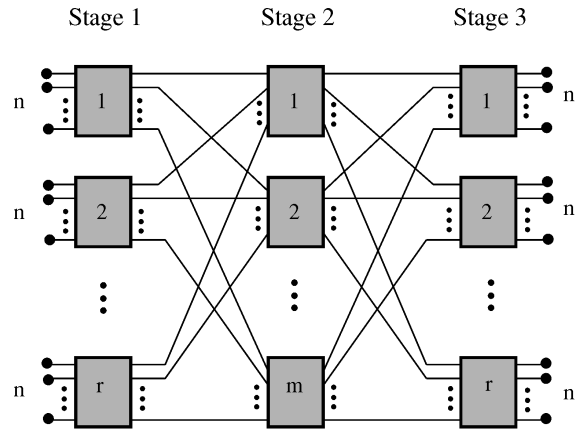

Fig. 1. A general schematic of a $v(m, n, r)$ network.

the minimum network latency this type of network can offer. This feature of constant stage networks is attractive for large scale highly parallel computing systems where communication delay is critical.

This type of network was first proposed by Clos [6]. The network has adjustable network parameters and can provide different type of connecting capabilities by choosing different values of the parameters. The general Clos network can have any odd number of stages and is built in a recursive fashion from smaller size networks. Therefore, it is sufficient to consider only the three-stage network. A three-stage Clos network with N input ports and N output ports has r switch modules of size $n \times m$ in stage 1, m switch modules of size $r \times r$ in stage 2, and r switch modules of size $m \times n$ in stage 3. The network has exactly one link between every two switch modules in its consecutive stages. Such a three-stage network is denoted as a $v(m, n, r)$ network. In a three-stage network, stage 1 is also referred to as *input stage*, stage 2 is also referred to as *middle stage*, and stage 3 is also referred to as *output stage*. A general schematic of a $v(m, n, r)$ network is shown in Fig. 1. We assume that every switch in the network has multicast capability, that is, each idle input link of a switch can be simultaneously connected to any subset of idle output links of the switch.

In general, the network cost of such a multistage network is measured by the number of crosspoints in the network. An $a \times b$ switch module is assumed to have ab crosspoints. It is easy to see that the network cost of a $v(m, n, r)$ network is proportional to the number of middle stage switches m for a fixed N and r.

Since output stage switches in a $v(m, n, r)$ network have multicast capability, a multicast connection can be described in terms of connections between an input port and output stage switches. The number of output stage switches in a multicast connection is referred to as the *fanout* of the multicast connection. Let O denote the set of all output stage switches. Based on the structure of the $v(m, n, r)$ network, we have $O = \{1, 2, \ldots, r\}$. For the ith input port in input stage, $i \in \{1, 2, \ldots, nr\}$, let $I_i \subseteq O$ denote the subset of the output stage switches to which input port i is to be connected in a multicast connection. I_i is referred to as an *input connection request* from input port i. Furthermore, if input port i can be connected to at most d ($1 \le d \le r$) output stage switches at a time (i.e., $|I_i| \le d$), we will refer to this input connection request as a *d-restricted input connection request*.

For a multicast assignment where each input switch can have at most α ($0 \le \alpha = \alpha(n, r) \le n$) input connection requests with unrestricted fanouts and all other input connection requests are d-restricted ($1 \le d \le r$), we will refer to it as an *(α, d)-multicast assignment*. Fig. 2 shows a (2, 1)-multicast assignment in a $v(5, 3, 4)$ network. We will refer to a $v(m, n, r)$ network that can realize all (α, d)-multicast assignments as a *$v_{\alpha,d}(m, n, r)$ multicast network*. Note that

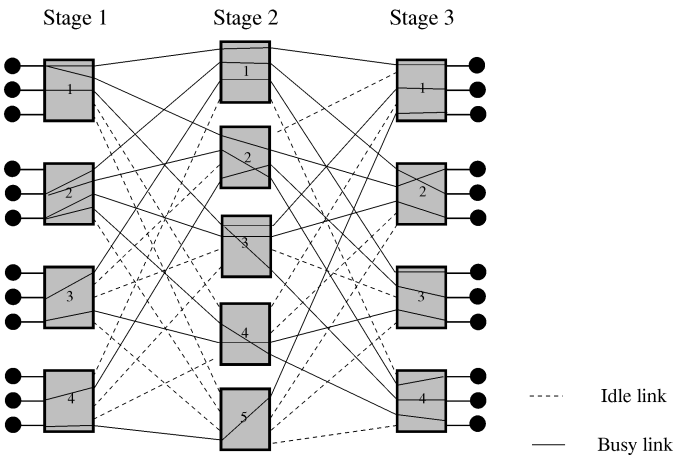

Fig. 2. A (2, 1)-multicast assignment in a $v(5, 3, 4)$ network.

in a $v_{\alpha,d}(m, n, r)$ network, those α multicast connections on each input stage switch are not tied to any specific subset of input ports and any input port can request an unrestricted multicast connection as long as the total number of unrestricted multicast connections on that input stage switch does not exceed α at that time. We will simply refer to a $v_{\alpha,1}(m, n, r)$ network as a $v_{\alpha}(m, n, r)$ network, where at most α input ports in each input stage switch can have unrestricted multicast connections at a time and all other input port can have only one-to-one connections. Clearly, a $v_n(m, n, r)$ network is a full multicast $v(m, n, r)$ network, and a $v_0(m, n, r)$ network is a classical permutation $v(m, n, r)$ network.

In addition, the multicast networks we consider in this paper are *nonblocking* networks in the sense that we can always satisfy an eligible multicast connection request without any *rearrangement* of existing connections in the network regardless of current network state. This eliminates the possible disruption of on-going communications caused by the rearrangements and the resulting time delay in path routings.

3 Previous Related Work

The $v(m, n, r)$ networks have been extensively studied in the literature [6], [7], [9], [10], [12], [13]. From the network structure described in Section 2, we know that two of the network parameters, n and r, are restricted by the network input/output size (in fact, $N = nr$), and the network cost is proportional to the number of middle stage switches m for a fixed N and r. Therefore, the main focus of the study has been on finding the minimum value of the network parameter m for a certain type of connecting capability to achieve the minimum network cost.

A recent design [12], [13] shows that a $v(m, n, r)$ network is nonblocking for arbitrary multicast assignments if the number of middle stage switches, m, satisfies $m \geq 3(n-1)\frac{\log r}{\log \log r}$. This result represents the currently best known design for constant stage nonblocking multicast networks. Furthermore, it is shown [14] that, under several typical routing control strategies, the necessary condition for a $v(m, n, r)$ multicast network to be nonblocking is $m \geq \Theta\left(n\frac{\log r}{\log \log r}\right)$, which matches the sufficient nonblocking condition for this type of network. However, it was shown [6], [7] that a $v(m, n, r)$ network is nonblocking for permutation assignments if $m \geq 2n - 1$.

Clearly, there is a considerably large gap in network cost between $v(m, n, r)$ multicast networks and $v(m, n, r)$ permutation networks. In the following, we will determine the nonblocking conditions for $v_{\alpha,d}(m, n, r)$ multicast networks. As we will see, $v_{\alpha,d}(m, n, r)$ networks compromise between full multicast networks and permutation networks: They have comparable cost to permutation networks and yet powerful enough multicast capability for multicast applications.

4 Nonblocking Conditions

In this section, we present the main results of this paper. We first give the nonblocking condition for general $v_{\alpha,d}(m, n, r)$ multicast networks. We then extend the result to yield the restricted multicast networks with the same order of network cost as $v(m, n, r)$ permutation networks.

Assume a $v_{\alpha,d}(m, n, r)$ network is currently providing some multicast connections from its input ports to its output ports. For any input port $i \in \{1, 2 \ldots, nr\}$, we will refer to the set of middle stage switches with currently unused links to the input switch associated with input port i as the *available middle switches*. Moreover, for any middle stage switch $j \in \{1, 2, \ldots, m\}$, we will refer to the subset of output stage switches to which middle switch j is providing connection paths from the input ports as the *destination set* of middle switch j and denote it as M_j. Clearly, we have $M_j \subseteq O$ for any $j \in \{1, 2, \ldots, m\}$. Notice that an output port can be connected to at most one input port at a time in a multicast connection. The following lemma reveals a global constraint to M_js.

LEMMA 1. *At any state of a $v_{\alpha,d}(m, n, r)$ multicast network, there are at most n* 1s, n 2s, …, n *rs distributed in the destination sets M_1, M_2, …, M_m.*

PROOF. Since any output stage switch k, $k \in \{1, 2, \ldots, r\}$, has n output ports, it can have at most n disjoint connection paths from the middle stage. This means that there are at most n ks in all destination sets $M_1, M_2, \ldots, M_m$. □

Now, given a new input connection request I_i, $i \in \{1, 2, \ldots, nr\}$, we need to find middle stage switches from the *available* middle switches to satisfy this connection request. The following lemma gives a necessary and sufficient condition for satisfying a connection request I_i.

LEMMA 2. *We can satisfy a connection request I_i using some x ($x \geq 1$) middle switches, say, $j_1, j_2, \ldots, j_x$, from among the available middle switches of a $v_{\alpha,d}(m, n, r)$ network if and only if $I_i \cap \left(\bigcap_{k=1}^{x} M_{j_k}\right) = \phi$.*

PROOF. If there exist x available middle switches, say, $j_1, j_2, \ldots, j_x$, for which $I_i \cap \left(\bigcap_{k=1}^{x} M_{j_k}\right) = \phi$, then, for every output switch t, $t \in I_i$, we can always find a middle switch, say j_k, $1 \leq k \leq x$, such that $t \notin M_{j_k}$, through which a connection path to output switch t is available. Thus, we can satisfy the new connection request through these x middle switches. Similarly, if we can satisfy connection request I_i using x middle switches, say, $j_1, j_2, \ldots, j_x$, then $I_i \cap \left(\bigcap_{k=1}^{x} M_{j_k}\right) = \phi$ before we satisfy this connection request. Otherwise, if there exists some output switch t, $t \in I_i \cap \left(\bigcap_{k=1}^{x} M_{j_k}\right)$, then a connection path could not be provided to output switch t through any middle switch in the set of x available middle switches. □

We now introduce a function which will be used in the proof of our first theorem. Let $\delta(n)$ be an integer function for any integer $n \geq 0$ satisfying

$$\delta(n) = \begin{cases} \frac{n}{2} - 1 & \text{if } n \text{ is even} \\ \left\lfloor \frac{n}{2} \right\rfloor & \text{if } n \text{ is odd} \\ 0 & \text{if } n = 0 \end{cases}$$

Denote $\delta^0(n) = n$, and $\delta^k(n) = \delta(\delta^{k-1}(n))$ for $k \geq 1$. Observe that for a sufficiently large k, $\delta^k(n) = 0$. We are interested in the smallest k such that $\delta^k(n) = 0$, and have the following lemma.

LEMMA 3. $\min\{k \mid \delta^k(n) = 0\} = \lfloor \log (n + 1) \rfloor$.

PROOF. See Appendix. □

We are now at the position to present our fundamental theorem.

THEOREM 1. *If there are at least $2n - 1$ available middle switches for a connection request with fanout f $(2 \leq f \leq r)$ in a $v_{\alpha,d}(m, n, r)$ network, we can always choose no more than $\lfloor \log (f + 1) \rfloor$ middle switches to satisfy this connection request from these available middle switches.*

PROOF. Without loss of generality, suppose the input connection request $I_i = \{1, 2, \ldots, f\}$, $2 \leq |I_i| = f \leq r$, and the destination sets of the available middle switches are $M_1, M_2, \ldots, M_{2n-1}$. By Lemma 1, there are at most $(n - 1)$ 1s, $(n - 1)$ 2s, …, $(n - 1)$ fs distributed among $M_1, M_2, \ldots M_{2n-1}$. Assign j_1 such that

$$\left|I_i \cap M_{j_1}\right| = \min_{1 \leq j \leq 2n-1} \left|I_i \cap M_j\right|.$$

Then we have

$$\left|I_i \cap M_{j_1}\right| \leq \frac{n-1}{2n-1} f.$$

Note that $\left|I_i \cap M_{j_1}\right|$ is an integer and $\frac{n-1}{2n-1} < \frac{1}{2}$. It follows that

$$\left|I_i \cap M_{j_1}\right| \leq \begin{cases} \frac{f}{2} - 1 & \text{if } f \text{ is even} \\ \left\lfloor \frac{f}{2} \right\rfloor & \text{if } f \text{ is odd} \end{cases}$$

which implies that

$$\left|I_i \cap M_{j_1}\right| \leq \delta(f).$$

Again, without loss of generality, suppose

$$I_i \cap M_{j_1} = \{1, 2, \ldots, f'\},$$

where $f' \leq \delta(f)$. Then, assign j_2 such that

$$\left|I_i \cap M_{j_1} \cap M_{j_2}\right| = \min_{\substack{1 \leq j \leq 2n-1 \\ j \neq j_1}} \left|I_i \cap M_{j_1} \cap M_j\right|.$$

Similarly, we have

$$\left|I_i \cap M_{j_1} \cap M_{j_2}\right| \leq \frac{n-1}{2n-1} \delta(f),$$

and this implies

$$\left|I_i \cap M_{j_1} \cap M_{j_2}\right| \leq \delta^2(f).$$

In general, in step k, we assign j_k such that

$$\left|I_i \cap M_{j_1} \cap M_{j_2} \cap \cdots \cap M_{j_k}\right| =$$

$$\min_{\substack{1 \leq k \leq 2n-1 \\ j \neq j_p, p<k}} \left|I_i \cap M_{j_1} \cap M_{j_2} \cap \cdots \cap M_{j_{k-1}} \cap M_j\right|$$

and

$$\left|I_i \cap M_{j_1} \cap M_{j_2} \cap \cdots \cap M_{j_k}\right| \leq \delta^k(f).$$

We are interested in the smallest integer k such that $\delta^k(f) = 0$. By Lemma 3, we have that

$$\min\{k \mid \delta^k(f) = 0\} = \lfloor \log (f + 1) \rfloor.$$

Therefore, there exists some $x \leq \lfloor \log (f + 1) \rfloor$ such that

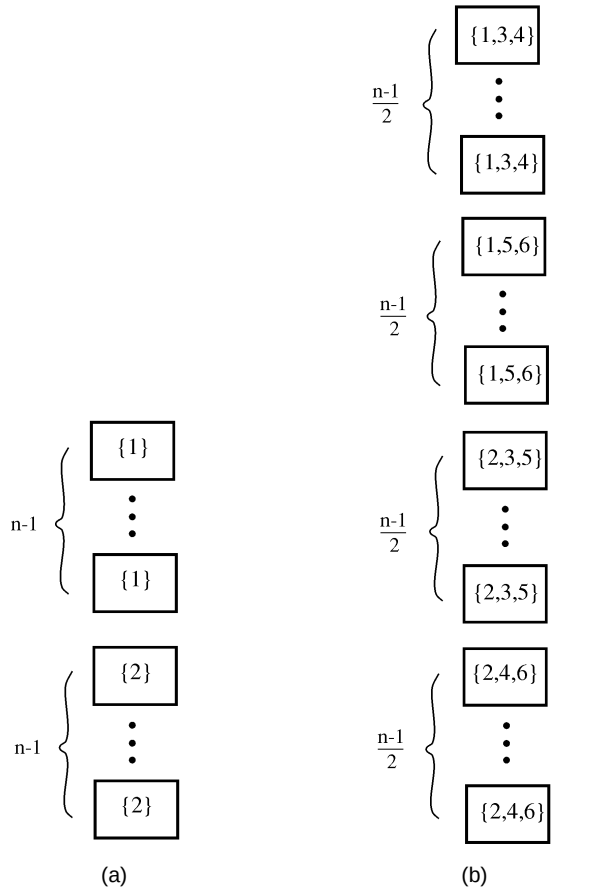

Fig. 3. The worst case network states of $2n - 2$ available middle switches for some connection requests. (a) $I_i = \{1, 2\}$. (b) $I_i = \{1, 2, 3, 4, 5, 6\}$.

$$\left|I_i \cap M_{j_1} \cap M_{j_2} \cap \cdots \cap M_{j_x}\right| = 0.$$

That is,

$$I_i \cap \left(\bigcap_{k=1}^{x} M_{j_k}\right) = \phi.$$

By Lemma 2, I_i can be satisfied by $M_{j_1}, M_{j_2}, \ldots, M_{j_x}$. □

We shall illustrate the above theorem by an example. Let $I_i = \{1, 2, 3, 4, 5, 6\}$ with $f = 6$. Among the $2n - 1$ available middle switches, there exists M_{j_1} such that $\left|I_i \cap M_{j_1}\right| \leq \frac{n-1}{2n-1} \times 6$, i.e., $\left|I_i \cap M_{j_1}\right| \leq 2$; then there exists M_{j_2} such that $\left|I_i \cap M_{j_1} \cap M_{j_2}\right| \leq \frac{n-1}{2n-1} \times 2$, i.e., $\left|I_i \cap M_{j_1} \cap M_{j_2}\right| = 0$. Therefore, the connection request I_i can be satisfied by no more than two (i.e., $\lfloor \log (6 + 1) \rfloor$) available middle switches.

Moreover, we can see that the number of available middle switches in Theorem 1 is tight. In other words, in some network states, $2n - 1$ available middle switches are necessary if we want to use no more than $\lfloor \log (f + 1) \rfloor$ middle switches to satisfy a connection request with fanout f. The following examples demonstrate that when only $2n - 2$ middle switches are available, we cannot choose no more than $\lfloor \log (f + 1) \rfloor$ middle switches to satisfy a connection request with fanout f.

Assume we have a connection request $I_i = \{1, 2\}$ ($f = |I_i| = 2$), and $2n - 2$ available middle switches. As shown in Fig. 3a, there exists a network state where there is a "1" in each destination set of $n - 1$ available middle switches and there is a "2" in each

destination set of the other $n-1$ available middle switches. This network state makes it impossible for us to choose one (i.e., $\lfloor \log (f+1) \rfloor$) middle switch among the $2n-2$ available middle switches to satisfy connection request I_i.

Now, we take a look at another example. Consider connection request $I_i = \{1, 2, 3, 4, 5, 6\}$ ($f = |I_i| = 6$), and $2n-2$ available middle switches. As show in Fig. 3b, there exists a network state where each destination set of the first $\frac{n-1}{2}$ available middle switches contains a subset {1, 3, 4}, each destination set of the second $\frac{n-1}{2}$ available middle switches contains a subset {1, 5, 6}, each destination set of the third $\frac{n-1}{2}$ available middle switches contains a subset {2, 3, 5}, and each destination set of the fourth $\frac{n-1}{2}$ available middle switches contains a subset {2, 4, 6}. It is easy to verify that there are a total of $(n-1)$ 1s, $(n-1)$ 2s, …, $(n-1)$ 6s distributed among these $2n-2$ available middle switches. Since the intersection of any two destination sets of these $2n-2$ available middle switches are not empty, we cannot choose no more than two (i.e., $\lfloor \log (f+1) \rfloor$) middle switches among these available middle switches to satisfy connection request I_i.

We have the following theorem regarding the nonblocking condition for general $v_{\alpha,d}(m, n, r)$ multicast networks.

THEOREM 2. *A $v_{\alpha,d}(m, n, r)$ multicast network ($d \geq 2$) is nonblocking if*

$$m \geq \alpha\left(\lfloor \log(r+1) \rfloor - \lfloor \log(d+1) \rfloor\right) + (n-1)(2 + \lfloor \log (d+1) \rfloor) + 1.$$

PROOF. We prove this theorem by considering the worst case network state: The new input connection request I_i has a fanout d and all other $n-1$ input ports on the same input switch as I_i are already connected to some output switches, among which α input ports have a fanout r and $(n-\alpha-1)$ input ports have a fanout d. Clearly, the middle switches providing connection paths for the other $n-1$ input ports on this input switch are not available for satisfying this new connection request. By Theorem 1, there are a total of $\alpha\lfloor \log (r+1) \rfloor + (n-\alpha-1)\lfloor \log (d+1) \rfloor$ middle switches not available to the new connection request. Also, by Theorem 1, if we still have $2n-1$ middle switches available, then we can satisfy the new connection request. In addition, this $2n-1$ available middle switches also guarantee that future connection requests from this input switch can always be satisfied. This is because after we satisfy I_i, we still have $2n-1-\lfloor \log (d+1) \rfloor$ available middle switches for any input port on this input switch and all input ports are connected to some output switches. Later, if any input port on this input switch wants to request a new connection, it must release the previous connection, which yields at least $\lfloor \log (d+1) \rfloor$ extra available middle switches. Therefore, in any case, we always have at least $2n-1$ available middle switches. By Theorem 1, we can satisfy any future connection request from this input switch. Similarly, we can apply the above argument to other input switches. Hence, the nonblocking condition for a $v_{\alpha,d}(m, n, r)$ network is

$$\begin{aligned} m &\geq \alpha\left\lfloor \log(r+1) \right\rfloor + (n-\alpha-1)\left\lfloor \log(d+1) \right\rfloor + 2n - 1 \\ &= \alpha\left(\left\lfloor \log(r+1) \right\rfloor - \left\lfloor \log(d+1) \right\rfloor\right) + (n-1)\left(2 + \left\lfloor \log(d+1) \right\rfloor\right) + 1. \end{aligned}$$

□

Next, we will discuss some interesting special cases of $v_{\alpha,d}(m, n, r)$ networks. Theorem 3 gives a more explicit nonblocking condition for a $v_{\alpha,d}$ network with certain α and d values.

THEOREM 3. *In a $v_{\alpha,2}(m, n, r)$ network, where $d \geq 2$, if at most $\frac{n\beta(r)}{\log r}$, where $1 \leq \beta(r) \leq \log r$, input ports on each input switch can have unrestricted multicast connections, and all other input ports can have multicast connections with fanout at most $2^{\beta(r)} - 1$, the nonblocking condition becomes $m \geq cn\beta(r)$, where c is a constant.*

PROOF. Setting $\alpha = \frac{n\beta(r)}{\log r}$ and $d = 2^{\beta(r)} - 1$ in Theorem 2, we have that

$$\begin{aligned} m &\geq \alpha\left(\left\lfloor \log(r+1) \right\rfloor - \left\lfloor \log(d+1) \right\rfloor\right) + (n-1)\left(2 + \left\lfloor \log(d+1) \right\rfloor\right) + 1 \\ &= \frac{n\beta(r)}{\log r}\left(\left\lfloor \log(r+1) \right\rfloor - \beta(r)\right) + (n-1)\left(2 + \beta(r)\right) + 1. \end{aligned}$$

Thus, there exists a constant c such that the network is nonblocking if $m \geq cn\beta(r)$. □

Now, let's look at an example of Theorem 3. Suppose that we let $\beta(r) = \log \log r$ in Theorem 3. Then, we have $\alpha = \frac{n \log \log r}{\log r}$ and $d = \log r - 1$. Therefore, the network is nonblocking if $m \geq 3n \log \log r$ for $r \geq 16$.

The analysis of the nonblocking condition for $v_\alpha(m, n, r)$ multicast networks is similar to that for $v_{\alpha,d}(m, n, r)$ networks except that, when $|I_i| = 1$, we can always choose one middle switch from n instead of $2n-1$ available middle switches. This is because when $|I_i| = 1$, say, $I_i = \{k\}$, $k \in \{1, 2, \ldots, r\}$, at most $n-1$ middle switches have k in their destination sets. We have the following theorem concerning the nonblocking condition for the $v_\alpha(m, n, r)$ networks.

THEOREM 4. *A $v_\alpha(m, n, r)$ multicast network is nonblocking if*

$$m \geq \alpha\lfloor \log(r+1) \rfloor + (2n - \alpha - 1) + (n - \lfloor \log (r+1) \rfloor)U_\alpha,$$

where,

$$U_\alpha = \begin{cases} 0 & \alpha = 0 \\ 1 & \alpha > 0 \end{cases}$$

PROOF. If the new input connection request is a one-to-one connection, i.e., $|I_i| = 1$, the worst case network state is that all other $n-1$ input ports on the same input switch as I_i are already connected to some output switches, among which α input ports have a fanout r and $(n-\alpha-1)$ input ports have a fanout 1. By Theorem 1, those α input ports with fanout r can occupy at most $\alpha\lfloor \log (r+1) \rfloor$ middle switches, and the remaining $(n-\alpha-1)$ input ports with fanout 1 occupy $(n-\alpha-1)$ middle switches. Therefore, there are a total of $\alpha\lfloor \log (r+1) \rfloor + (n-\alpha-1)$ middle switches not available to the new connection request. To guarantee that we can always satisfy this new one-to-one connection request, we need at least n available middle switches. Hence, the nonblocking condition is

$$m \geq \alpha\lfloor \log (r+1) \rfloor + (n-\alpha-1) + n.$$

On the other hand, if the new input connection request has a fanout $|I_i| > 1$, the worst case network state is that all other $n-1$ input ports on the same input switch as I_i are already connected to some output switches, among which $\alpha - 1$ input ports have a fanout r and $(n-\alpha)$ input ports have a fanout 1. Then, by Theorem 1, to guarantee that we can always satisfy this new connection request, we need at least $2n-1$ available middle switches. This leads to the nonblocking condition

$$m \geq (\alpha-1)\lfloor \log (r+1) \rfloor + (n-\alpha) + 2n - 1.$$

Combining these two cases, we obtain the nonblocking condition stated in the theorem. □

TABLE 1
NONBLOCKING CONDITIONS FOR SEVERAL TYPICAL RESTRICTED MULTICAST NETWORKS

Network	Permutation $v(m,n,r)$	$v_\alpha(m,n,r)$ $\alpha=1$	$v_\alpha(m,n,r)$ $\alpha=\frac{0.5n}{\log r}$	$v_{\alpha,d}(m,n,r)$ $\alpha=\frac{n}{\log r}$ $d=2$	$v_{\alpha,d}(m,n,r)$ $\alpha=\frac{n\log\log r}{\log r}$ $d=\log r$	Full multicast $v(m,n,r)$
Nonblocking condition m	$2n-1$	$3n-2$	$3.5n-3$	$4n-3$	$3n\log\log r$	$3n\frac{\log r}{\log\log r}$
Unrestricted multicast ports per input switch	0	1	$\frac{0.5n}{\log r}$	$\frac{n}{\log r}$	$\frac{n\log\log r}{\log r}$	n

It is easy to see that the special case $\alpha = 0$ in Theorem 4 gives $m \geq 2n-1$ which matches the nonblocking condition for $v(m, n, r)$ permutation networks.

We are particularly interested in the restricted multicast networks with $\alpha > 0$ which have the same order of network cost as permutation networks. Theorem 5 gives the nonblocking condition for such networks.

THEOREM 5. *In a $v_{\alpha,d}(m, n, r)$ network, if at most $\frac{c_1 n}{\log r}$ input ports on each input switch can have unrestricted multicast connections, and all other input ports can have multicast connections with fanout c_2, where c_1 and c_2 are constants, the nonblocking condition becomes $m \geq cn$, where c is a constant.*

PROOF. First, when the restricted fanout $d = c_2 = 1$, we set $\alpha = \frac{c_1 n}{\log r}$ in Theorem 4 and obtain

$$m \geq \frac{c_1 n}{\log r}\lfloor \log(r+1)\rfloor + 3n - \frac{c_1 n}{\log r} - 1 - \lfloor \log(r+1)\rfloor.$$

Thus, there exists a constant c such that $m \geq cn$.

When the restricted fanout $d = c_2 \geq 2$, we set $\alpha = \frac{c_1 n}{\log r}$ and $d = c_2$ in Theorem 2, and obtain

$$m \geq \frac{c_1 n}{\log r}\left(\lfloor \log(r+1)\rfloor - \lfloor \log(c_2+1)\rfloor\right) + (n-1)\left(2 + \lfloor \log(c_2+1)\rfloor\right) + 1.$$

Thus, there also exists a constant c, such that $m \geq cn$. □

Recall that the nonblocking condition for the $v(m, n, r)$ permutation network is $m \geq 2n-1$. Since the network cost is proportional to the number of middle switches, m, it is easy to see that the $v_{\alpha,d}(m, n, r)$ networks that satisfy the condition in Theorem 5 have the same order of network cost as permutation networks. Theorem 5 suggests that, at any time, each input switch can have up to $\frac{cn}{\log r}$ input ports out of its n input ports making unrestricted multicast connections and the remaining input ports making constant fanout multicast connections while keeping the network cost comparable to a permutation network. Under this nonblocking condition, the number of input ports that can request unrestricted multicast connections at a time are generally adequate for many multicast applications. For example, in a parallel computing system, we can consider all processors connected to an input switch as a cluster which are cooperating to complete a common task. At any given time, not all processors in the cluster need to perform full multicast, and we can have up to $\frac{cn}{\log r}$ processors in the cluster performing full multicast. Moreover, the only thing the higher-level software and algorithm designers need to be concerned is to keep the number of processors performing full multicasting in the cluster below the threshold $\frac{cn}{\log r}$. This is a fairly simple rule for judging whether an arbitrary multicast connection can be realized in a single pass through the network.

Finally, we summarize the nonblocking conditions for several typical $v_{\alpha,d}(m, n, r)$ networks along with permutation $v(m, n, r)$ network and full multicast $v(m, n, r)$ network in Table 1.

From Table 1, we can see that the newly designed restricted multicast networks can realize a substantial number of well-defined multicast assignments while keeping network cost comparable to $v(m, n, r)$ permutation networks. Moreover, the multicast capability of the networks is represented as a function of fundamental network structural parameters so that the trade-off between the network multicast capability and the network cost can be determined. This enables different system designers to choose the multicast networks which fit in their particular application needs.

5 THE ROUTING ALGORITHM

In this section, we present a routing algorithm for satisfying connection requests in a $v_{\alpha,d}(m, n, r)$ network.

Given a $v_{\alpha,d}(m, n, r)$ network satisfying the nonblocking condition in Theorem 2 or Theorem 4 and an input connection request I_i. Then, there are at least k available middle switches for I_i where $k = 2n-1$ if $|I_i| \geq 2$, or $k = n$ if $|I_i| = 1$. Take any k middle switches from these available middle switches. Without loss of generality, let these available middle switches be $M_1, M_2, \ldots, M_k$. Let $A[j]$ $(1 \leq j \leq r)$ denote the number of input connections with fanout greater than d in the jth input switch. Table 2 shows the routing algorithm for connecting I_i.

We now give some necessary explanations for the routing algorithm in Table 2. In the algorithm, *MASK* stores a subset of I_i which has not yet been assigned to any available middle switches at current execution time. S stores the indexes of the selected middle switches to satisfy the input connection request I_i, and $H[p]$ stores a subset of I_i which will be realized by middle switch p. The first while loop in the algorithm is to find middle switches to satisfy the connection request I_i. From Theorems 1, 2, and 4, we know that at most $\max\{1, \log |I_i|\}$ middle switches are needed for satisfying I_i. At the end of the first while loop, S stores the indexes of selected middle switches which together will satisfy I_i. In fact, we can show that, at the end of the first while loop, the following conditions hold:

1) for any $p \in S$, $H[p] \cap M_p = \phi$;
2) for any $p, q \in S$, and $p \neq q$, $H[p] \cap H[q] = \phi$;
3) $I_i = \bigcup_{p \in S} H[p]$.

Therefore, I_i can be distributed to the set of middle switches indexed by the elements of S. This is accomplished in the second while loop of the algorithm. In other words, set $H[p]$ is distributed to middle switch p for all $p \in S$ in the second while loop.

The example in Fig. 4 shows how the routing algorithm works. For $n = r = 8$, given a connection request $I_i = \{1, 2, 3, 4, 5, 6, 7, 8\}$ with fanout $f = 8$, and $k = 2n - 1 = 15$ available middle switches with destination sets $M_1, M_2, \ldots, M_{15}$ shown in Fig. 4a. In the first iteration, *MASK* is equal to the full set $\{1, 2, \ldots, 8\}$, and we directly compare the cardinalities of all M_js for $1 \leq j \leq 15$. Since all cardinalities in this example are the same, we randomly choose one

$$
\begin{array}{lll}
MASK = \{1,2,3,4,5,6,7,8\} & MASK = \{1,2,3\} & MASK = \{1\} \quad MASK = \phi \\
\Rightarrow \quad M_1 = \{1,2,3\}, H_1 = \{4,5,6,7,8\} & M'_1 = \{1,2,3\} & M''_1 = \{1\} \\
M_2 = \{1,2,3\} & M'_2 = \{1,2,3\} & M''_2 = \{1\} \\
M_3 = \{1,2,3\} & M'_3 = \{1,2,3\} & M''_3 = \{1\} \\
M_4 = \{1,4,5\} & \Rightarrow \quad M'_4 = \{1\}, H_4 = \{2,3\} & M''_4 = \{1\} \\
M_5 = \{1,4,5\} & M'_5 = \{1\} & M''_5 = \{1\} \\
M_6 = \{1,4,5\} & M'_6 = \{1\} & M''_6 = \{1\} \\
M_7 = \{1,6,7\} & M'_7 = \{1\} & M''_7 = \{1\} \\
M_8 = \{2,4,6\} & M'_8 = \{2\} & \Rightarrow \quad M''_8 = \phi, H_8 = \{1\} \\
M_9 = \{2,4,6\} & M'_9 = \{2\} & M''_9 = \phi \\
M_{10} = \{2,5,7\} & M'_{10} = \{2\} & M''_{10} = \phi \\
M_{11} = \{2,5,7\} & M'_{11} = \{2\} & M''_{11} = \phi \\
M_{12} = \{3,4,7\} & M'_{12} = \{3\} & M''_{12} = \phi \\
M_{13} = \{3,4,7\} & M'_{13} = \{3\} & M''_{13} = \phi \\
M_{14} = \{3,5,6\} & M'_{14} = \{3\} & M''_{14} = \phi \\
M_{15} = \{3,5,6\} & M'_{15} = \{3\} & M''_{15} = \phi \\
\text{(a)} & \text{(b)} & \text{(c)}
\end{array}
$$

Fig. 4. Routing example for connection request I_i = {1, 2, 3, 4, 5, 6, 7, 8}.

from them, say, M_1. Then, H_1 = {4, 5, 6, 7, 8} stores the portion of I_i which is assigned to middle switch 1. In the second iteration, *MASK* becomes {1, 2, 3} which is the set of remaining elements in I_i. Denote $M_j \cap MASK$ as M'_j (see Fig. 4b.) By comparing the cardinalities of M'_js, we randomly choose one from them with the minimum cardinality, say, M'_4. Then, H_4 = {2, 3} stores the portion of I_i which is assigned to middle switch 4. In the third iteration, *MASK* becomes {1}. Denote $M'_j \cap MASK$ as M''_j (Fig. 4c). As can be seen, those empty M''_js clearly have the minimum cardinality. We choose one of them, say M''_8. Then, H_8 = {1} is the portion of I_i assigned to middle switch 8. Finally, *MASK* becomes empty, three (i.e., $\lfloor \log (f + 1) \rfloor$) available middle switches 1, 4, and 8 are chosen for satisfying connection request I_i.

TABLE 2
THE ROUTING ALGORITHM FOR $v_{\alpha',d}(m, n, r)$ NETWORKS

Algorithm:
/* Check the eligibility of the connection request and update $A[j]$. */
$j = \lceil \frac{i}{n} \rceil$;
if $(|I_i| > d)$ **then** {
  if $(A[j] < \alpha)$ **then**
    $A[j] = A[j] + 1$;
  else exit without making connection;
}
/* Find up to $x = \max\{1, \log |I_i|\}$ middle switches for I_i */
$MASK \leftarrow I_i$;
$S \leftarrow \phi$;
$T \leftarrow \{1, 2, \ldots, k\}$;
for $i = 1$ **to** x **do**
 $H[i] \leftarrow \phi$;
while $(MASK \neq \phi)$ {
  choose middle switch p such that
  $|M_p \cap MASK| = \min_{q \in T} |M_q \cap MASK|$;
  $S \leftarrow S \cup \{p\}$;
  $T \leftarrow T - \{p\}$;
  $H[p] \leftarrow MASK - M_p$;
  $MASK \leftarrow M_p \cap MASK$;
}
/* Distribute I_i to the selected middle switches in S.*/
while $(S \neq \phi)$ {
  take $p \in S$;
  $M_p \leftarrow M_p \cup H[p]$;
  $S \leftarrow S - \{p\}$;
}
End

We now analyze the complexity of the above algorithm. The time for one iteration of the first while loop is proportional to $|MASK| \cdot |T|$. Since the number of available middle switches is $k = O(n)$, after each iteration, $|MASK|$ reduces its value to half. We know that initially $|MASK| = |I_i| \leq r$ and $|T| = k$. Thus, the total time for the first while loop is proportional to $|I_i| \cdot k$, i.e., $O(N)$. Clearly, the second while loop also takes $O(N)$ time. The rest of the algorithm takes less than $O(N)$ time. Thus, the time complexity of the above algorithm is linear to the network size N. Moreover, by employing the techniques used in [13], we can obtain a parallel routing algorithm for the above routing process with time complexity of $O(\log^2 r)$.

6 CONCLUSIONS

In this paper, we have presented a class of interconnection networks for supporting multicast communications in parallel computing systems. The newly designed networks can support a substantial number of well-defined multicast assignments in a nonblocking fashion and still keep the same order of network cost as permutation networks. We have also presented an efficient routing algorithm for satisfying connection requests in such networks. Moreover, the multicast capability of the networks is represented as a function of fundamental network structural parameters so that the trade-off between the network multicast capability and the network cost can be determined.

APPENDIX

PROOF OF LEMMA 3. Consider three consecutive integers $2n + 1$, $2n$, and $2n - 1$. We have

$$\delta(2n + 1) = n,\ \delta(2n) = n - 1, \text{ and } \delta(2n - 1) = n - 1.$$

In other words, for any integers $n_1 \geq n_2$, we always have $\delta(n_1) \geq \delta(n_2)$, which means that function $\delta(n)$ is a nondecreasing function.

Define $z(n) = \min\{k \mid \delta^k(n) = 0\}$. Clearly, $z(1) = z(2) = 1$. Also note that

$$\delta(2^i - 1) = 2^{i-1} - 1 \text{ for } i \geq 1, \text{ and } \delta(2^i - 2) = 2^{i-1} - 2 \text{ for } i \geq 2.$$

Then,

$$z(2^i - 1) = \min\{k \mid \delta^k(2^i - 1) = 0\} = 1 + z(2^{i-1} - 1)$$
$$= \cdots = (i - 1) + z(2^1 - 1) = i,$$

and

$$z(2^i - 2) = \min\{k \mid \delta^k(2^i - 2) = 0\} = 1 + z(2^{i-1} - 2)$$
$$= \cdots = (i - 2) + z(2^2 - 2) = i - 1.$$

Since $z(\cdot)$ is nondecreasing, for any given integer n satisfying

$$2^i - 1 \le n \le 2^{i+1} - 2, \quad (1)$$

where $i \ge 1$, we have that

$$i = z(2^i - 1) \le z(n) \le z(2^{i+1} - 2) = i.$$

That is,

$$z(n) = i. \quad (2)$$

On the other hand, (1) is equivalent to

$$2^i \le n + 1 \le 2^{i+1} - 1 < 2^{i+1},$$

it follows that

$$i \le \log(n + 1) < i + 1.$$

Hence,

$$i = \lfloor \log (n + 1) \rfloor. \quad (3)$$

Combining (2) and (3), we obtain $z(n) = \lfloor \log (n + 1) \rfloor$. □

ACKNOWLEDGMENTS

This research was supported in part by the U.S. Army Research Office under Grant No. DAAH04-96-1-0234 and by the U.S. National Science Foundation under Grant No. MIP-9522532 and OSR-9350540.

REFERENCES

[1] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, Mar. 1994.
[2] J. Duato, S. Yalmanchili, and L.M. Ni, *Interconnection Networks: An Engineering Approach*. Los Alamitos, Calif.: IEEE CS Press, 1997.
[3] L.M. Ni and D.K. Panda, "A Report of the ICPP '94 Panel on Sea of Interconnection Networks: What's Your Choice?" *IEEE CS Technical Committee Computer Architecture Newsletter,* pp. 31-44, 1994.
[4] L.M. Ni, "Should Scalable Parallel Computers Support Efficient Hardware Multicast?" *Proc. ICPP'95 Workshop Challenges for Parallel Processing*, pp. 2-7, 1995.
[5] D.K. Panda, "Issues in Designing Efficient and Practical Algorithms for Collective Communication on Wormhole-Routed Systems," *Proc. ICPP'95 Workshop Challenges for Parallel Processing*, pp. 8-15, 1995.
[6] C. Clos, "A Study of Non-Blocking Switching Networks," *The Bell System Technical J.*, vol. 32, pp. 406-424, 1953.
[7] V.E. Benes, "Heuristic Remarks and Mathematical Problems Regarding the Theory of Switching Systems," *The Bell System Technical J.*, vol. 41, pp. 1,201-1,247, 1962.
[8] G.W. Richards and F. K. Hwang, "A Two-Stage Rearrangeable Broadcast Switching Network," *IEEE Trans. Comm.*, vol. 33, pp. 1,025-1,034, 1985.
[9] G.M. Masson and B.W. Jordan, "Generalized Multi-Stage Connection Networks," *Networks*, vol. 2, pp. 191-209, 1972.
[10] F.K. Hwang and A. Jajszczyk, "On Nonblocking Multiconnection Networks," *IEEE Trans. Comm.*, vol. 34, pp. 1,038-1,041, 1986.
[11] P. Feldman, J. Friedman, and N. Pippenger, "Wide-Sense Nonblocking Networks," *SIAM J. of Discrete Math.*, vol. 1, no. 2, pp. 158-173, 1988.
[12] Y. Yang and G.M. Masson, "Nonblocking Broadcast Switching Networks," *IEEE Trans. Computers*, vol. 40, no. 9, pp. 1,005-1,015, Sept. 1991.
[13] Y. Yang and G.M. Masson, "Fast Path Routing Techniques for Nonblocking Broadcast Networks," *Proc. 13th IEEE Int'l Phoenix Conf. Computers and Comm.*, pp. 358-364, 1994.
[14] Y. Yang and G.M. Masson, "The Necessary Conditions for Clos-Type Nonblocking Multicast Networks," *Proc. 10th Int'l Parallel Processing Symp.*, pp. 789-795, 1996.
[15] J.P. Ofman, "A Universal Automation," *Trans. Moscow Math. Soc.*, vol. 14, 1965 (translation published by *Am. Math. Soc.* pp. 1,119-1,125, 1967).
[16] C.D. Thompson, "General Connection Networks for Parallel Processor Interconnection," *IEEE Trans. Computers*, vol. 27, pp. 1,119-1,125, 1978.
[17] C.-T. Lea, "A New Broadcast Switching Network," *IEEE Trans. Comm.*, vol. 36, pp. 1,128-1,137, 1988.
[18] C. Lee and A.Y. Oruc, "Design of Efficient and Easily Routable Generalized Connectors," *IEEE Trans. Comm.*, vol. 43, pp. 646-650, 1995.
[19] Y. Yang and G.M. Masson, "Broadcast Ring Sandwich Networks," *IEEE Trans. Computers*, vol. 44, no. 10, pp. 1,169-1,180, Oct. 1995.