The Necessary Conditions for Clos-Type Nonblocking Multicast Networks

Yuanyuan Yang, *Senior Member, IEEE*, and Gerald M. Masson, *Fellow, IEEE*

Abstract—Efficient interconnection networks are critical in providing low latency, high bandwidth communication in parallel and distributed computing systems with hundreds or thousands of processors. The well-known Clos network or $v(m,n,r)$ network can be extended to provide full one-to-many or multicast capability. In this paper, we consider several typical routing control strategies for Clos-type nonblocking multicast networks and derive the necessary conditions under which this type of network is nonblocking for arbitrary multicast assignments in the strict sense as well as under these control strategies. The necessary conditions obtained are represented as the number of middle stage switches $m \geq \Theta\left(n\frac{\log r}{\log\log r}\right)$. These results match the sufficient nonblocking condition for the currently best available explicitly constructed, constant stage nonblocking multicast network [8], [9], and provide a basis for the optimal design of this type of multicast network.

Index Terms—Interconnection networks, multicast networks, routing control strategies, nonblocking, necessary conditions.

1 Introduction

Efficient interconnection networks are critical in providing low latency, high bandwidth communication in parallel and distributed computing systems with hundreds or thousands of processors. In many communication environments, we often need to support *one-to-many* connections, where one source is sending the same message to multiple destinations. One-to-many connections are also called *multicast*. In a *multicast connection* through a multistage network, an input port can be simultaneously connected to more than one output port, but an output port can be connected to at most one input port at a time. A *multicast network* we consider in this paper is a multistage network which can realize all possible simultaneous nonoverlapped multicast connections between the network input ports and the network output ports. A *nonblocking* multicast network is a multicast network which can realize all new multicast connections without any rearrangements of existing connections in the network, regardless of the current network state. Due to the possible disruption of ongoing communication caused by the rearrangements, as well as the resulting time delay in path routings, nonblocking capability is generally desirable.

Supporting multicast in parallel and distributed computers has become an increasingly important issue [1], [2]. There has been a lot of work on multicast networks in the literature [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. In this paper, we are mainly concerned with a class of explicitly constructed nonblocking multicast network based on Clos network [19], [20]. Clos-type networks have been widely used in various interconnection problems. Some recent applications include the NEC ATOM switch designed for BISDN [21], the IBM GF11 multiprocessor [22], and ANSI Fibre Channel Standard for interconnection of processors to the I/O system. More recently, it was shown [23] that the network in the IBM SP2 is functionally equivalent to the Clos network. 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 generally sufficient to consider only the three-stage network. In general, a three-stage Clos network with N input ports and N output ports (i.e., an $N \times N$ network) 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 multistage network is denoted as a $v(m,n,r)$ network. In three-stage networks, stage 1, stage 2, and stage 3 are also referred to as *input stage, middle stage, and output stage,* respectively. A general schematic of a $v(m,n,r)$ network is shown in Fig. 1. Based on the three-stage network described above, for any $k \geq 1$, we can construct a $(2k+1)$-stage $N \times N$ network by replacing each $r \times r$ switch at the middle stage in a $v(m,n,r)$ network with a $(2k-1)$-stage $r \times r$ network.

Several designs have been proposed for this type of multicast network [3], [5], [8], [9]. Since two of the $v(m,n,r)$ network parameters, n and r, are restricted by the network input/output size N, the main focus of the study is to find the minimum value of the network parameter m for multicast capability to achieve the minimum network cost. By employing a special control strategy for choosing the middle stage switches for satisfying connection requests, the most recent design [8], [9] gave the currently best available sufficient condition for a $v(m,n,r)$ network, $m \geq cn\frac{\log r}{\log\log r}$, where c is a small constant, which guarantees that the network is nonblocking for arbitrary multicast

- *Y. Yang is with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794. E-mail: yang@ece.sunysb.edu.*
- *G.M. Masson is with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218. E-mail: masson@cs.jhu.edu.*

Manuscript received 27 June 1996.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 102047.

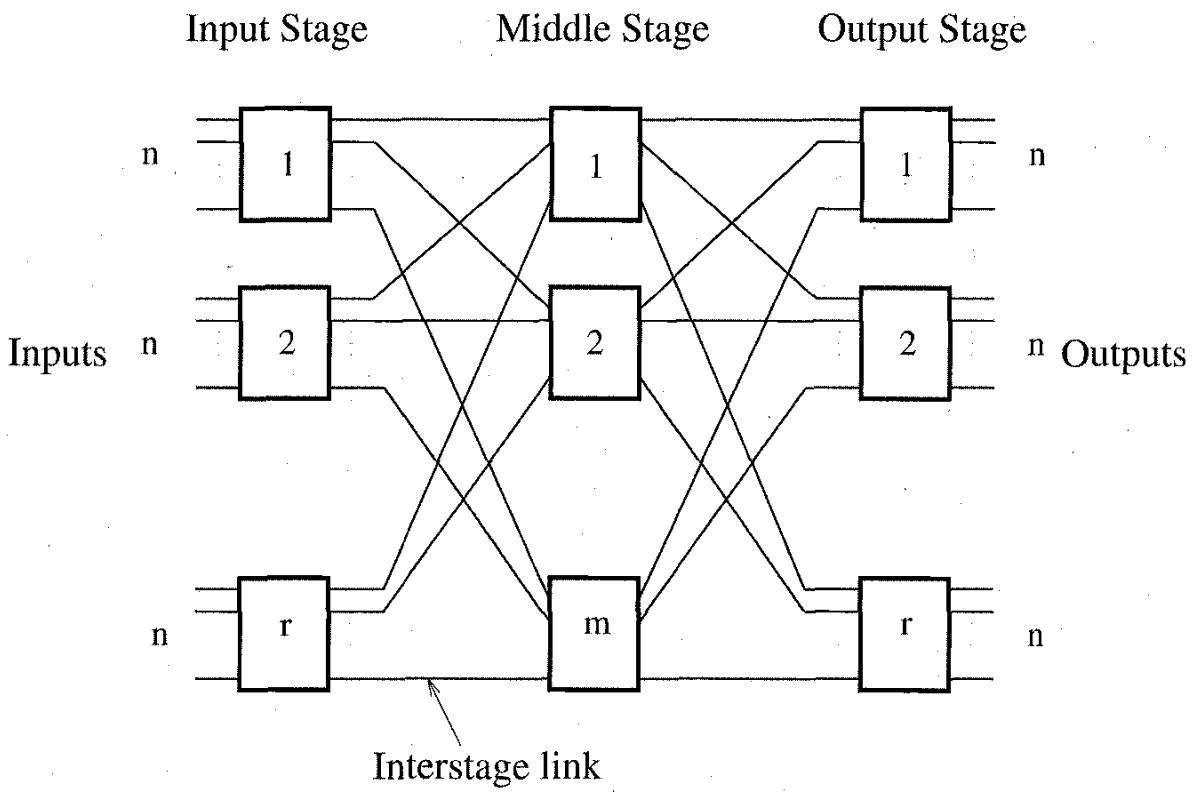

Fig. 1. A general schematic of a $v(m,n,r)$ network.

connections. This result yields an explicitly constructed, constant stage nonblocking multicast network with lowest network cost.

However, it is still unknown whether the sufficient condition given in [8] can be further reduced and what the optimal design for this type of network is. Apparently, there are some network routing control strategies which seem to be promising for further reducing the network cost of the $v(m,n,r)$ multicast network. In this paper, we consider several typical such control strategies, and analyze nonblocking conditions under these control strategies. In contrast to the previous design [8], [9], we will approach this problem from another angle: exploring the necessary conditions on the number of middle stage switches required for nonblocking in a $v(m,n,r)$ multicast network. These necessary conditions will provide a basis for the optimal design of multicast networks. We will consider the necessary nonblocking conditions in general, as well as under these typical routing control strategies. The necessary conditions will be derived by constructing some worst case network states which require a certain number of middle stage switches.

The rest of the paper is organized as follows: In Section 2, some basic definitions and observations will be given. In Section 3, several network control strategies will be described and compared. The main results of the paper will be presented in Section 4. In Section 5, complexity of the control strategies and implementation issues will be discussed. The conclusion will be given in the last section.

2 Definitions and Observations

In this section, we give some basic definitions and observations which are used in this paper.

First, it is reasonable to assume that every switch in the $v(m,n,r)$ multicast network has multicast capability. Recall that a $v(m,n,r)$ network can be recursively constructed; that is, each switch requiring multicast capability can be replaced by a multicast network of the same size. Thus, eventually, we may only need to build multicast capability into very small switches, such as 2×2 or 4×4 switches. This involves only minor increases in complexity, even in electronic switches, and, in optical switches, the complexity increases are even smaller.

Since output stage switches have multicast capability, a multicast connection can therefore be described in terms of connections between an input port and output stage switches. 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 switch modules in the output stage 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. Then, a *multicast assignment* which represents a maximal set of nonoverlapped multicast connections between the input ports and the output ports can be characterized by a set

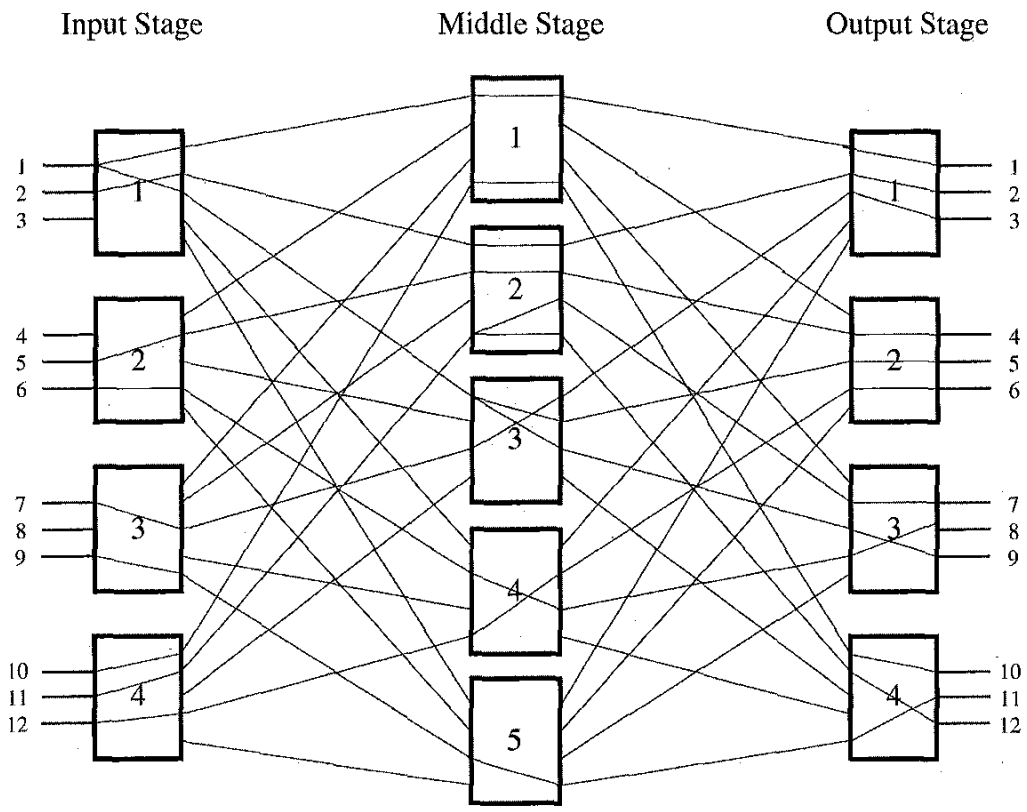

Fig. 2. A multicast assignment in a $v(5,3,4)$ network.

$$B = \{I_1, I_2, \ldots, I_{nr}\}$$

For example, for the multicast assignment shown in Fig. 2, we have that

$$B = \{\{1,2,3\},\{1\},\{\},\{\},\{2\},\{3\},\{1\},\{\},\{4\},\{4\},\{3,4\},\{2\}\}.$$

Also, notice that, since an output port can be connected to at most one input port at a time in a $v(m,n,r)$ multicast network, there is a global constraint to set B: There are at most n 1s, n 2s, ... , n rs distributed in B. Thus, the flattened set of B is a multiset chosen from an r-element set $\{1,2,\ldots,r\}$ with the multiplicity of each element no more than n (by the flattened set of a set, we mean the set obtained by removing all internal brackets in the original set). This property of the $v(m,n,r)$ networks is important for deriving the necessary nonblocking conditions in later sections.

To characterize the connection state between the input stage and middle stage in a $v(m,n,r)$ network, 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 stage switch associated with input port i as the *available middle switches*. For example, for the multicast assignment shown in Fig. 2, the available middle switch of input port 3 is $\{4,5\}$ and the available middle switches of input port 8 is $\{1,2,4\}$.

To characterize the state of m switches in the middle stage of a $v(m,n,r)$ network, let $M_j \subseteq O, j \in \{1,2,\ldots,m\}$, denote the subset of outputs of middle stage switch j which are currently occupied. M_j is referred to as the *destination set* of middle stage switch j. Then, the state of the middle stage of a $v(m,n,r)$ network can be denoted as

$$M = \{M_1, M_2, \ldots, M_m\}.$$

For example, for the multicast assignment shown in Fig. 2, we have

$$M = \{\{1,4\},\{1,2,3,4\},\{1,2,3\},\{2,3\},\{4\}\}.$$

Since all connections in a multicast assignment must go through some middle stage switches, it should be clear that, in general, for any state of a $v(m,n,r)$ network, the flattened set of M is equal to the flattened set of B. Thus, the problem of designing $v(m,n,r)$ multicast networks can be considered as the problem of distributing all elements in any multicast assignment set B to the minimum number of sets $M_1, M_2, \ldots, M_m$ such that, for all $j = 1,2,\ldots,m$,

$$M_j = \bigcup_{i=1}^{r} C_{ij} \subseteq \{1,2,\ldots,r\}, \tag{1}$$

where $C_{ij} \subseteq I_{i_j}$, $i_j \in \{(i-1)n+1, (i-1)n+2, \ldots, in\}$, is the contribution of input stage switch i to M_j. Notice that we may have different C_{ij}s in (1) under different *network routing control strategies*, which will be discussed in the next section.

3 Network Routing Control Strategies

In general, a network routing control strategy specifies the setting control of switch modules in a network. In particular, in a $v(m,n,r)$ network, how to choose middle stage switches for satisfying a connection request is the main concern of a control strategy.

Assume a $v(m,n,r)$ network is currently providing some multicast connections from its input ports to its output ports. The middle stage state is represented by the destination sets of middle stage switches, that is,

$$M = \{M_1, M_2, \ldots, M_m\}.$$

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:

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

In general, there are many ways to choose middle stage switches among available middle switches for satisfying a connection request and some control strategies can be employed to make the decision. Different control strategies set up different criteria for choosing middle stage switches and may have different effects on the total number of middle stage switches required to guarantee a network to be nonblocking. Also, the implementation complexity of control strategies may vary. We will have more discussions on this issue in Section 5. In particular, no control strategy is employed to govern the process of choosing middle stage switches can be viewed as a special control strategy which yields the strictly nonblocking network. Due to the nonuniform nature of multicast connections, if no control strategy is employed, we can expect that the number of middle stage switches required for nonblocking becomes very large. Therefore, we must employ some kind of "intelligent" routing control strategy which carefully selects the paths used to satisfy the current connection request so that the nonblocking connecting capability for future potential multicast connection requests can be maintained and, at the same time, the number of middle stage switches are kept small. This type of nonblocking capability can be referred to as *control-strategy-based nonblocking* capability or *wide-sense nonblocking* capability. It is this type of nonblocking capability that we consider in this paper. In the following, we will describe several typical control strategies for choosing middle stage switches from the available middle switches in a $v(m,n,r)$ multicast network.

Strategy 1. *For each input connection request, I_i, $i \in \{1,2,\ldots,nr\}$, in the network, always choose the middle stage switch with the minimum cardinality of destination sets with regard to the unsatisfied portion of I_i from available middle switches until I_i is satisfied, that is, until all middle stage switches chosen satisfy the condition in Lemma 1.*

Strategy 1 is the control strategy employed in [8]. It was shown in [8] that if there are at least $nr^{1/x}$ ($1 \leq x \leq \min\{n-1, r\}$) available middle switches, we can always choose no more than x middle stage switches to satisfy I_i. This yields that the number of middle stage switches $m = O\left(n\frac{\log r}{\log\log r}\right)$. A linear network routing algorithm was also provided [8] which repeatedly does finding of smallest cardinality and intersections among the destination sets of available middle switches to implement this control strategy. A parallel routing algorithm was presented [9] to implement the same control strategy to further speed up the path routing process.

Strategy 2. *For each input connection request I_i, $i \in \{1,2,\ldots,nr\}$, choose the minimum number of middle stage switches that satisfy the condition in Lemma 1 for the current network state from the available middle switches.*

Notice the difference between Strategy 2 and Strategy 1. Instead of choosing middle stage switches in the order of smaller destination set cardinality first for an input connection request in Strategy 1, Strategy 2 always finds the minimum number of available middle switches for the current input connection request. In other words, Strategy 2 achieves "local optimization" within a single input stage switch by using as few as possible output links in the input stage switch and keeping the maximum number of available middle switches for future connection requests from that input stage switch. For example, as shown in Fig. 3a, suppose we have a connection request from input port 2 to output ports 2, 5, 7, and 10 (in terms of output stage switches, $I_2 = \{1,2,3,4\}$), and input port 2 has three available middle switches: middle stage switches 1, 2, and 3 with destination sets $M_1 = \{1,4\}$, $M_2 = \{1,2\}$, and $M_3 = \{3,4\}$, respectively. Consider Strategy 1 first. Note that the cardinalities of M_1, M_2, and M_3 all are equal to 2. Based on the routing algorithm in [8], we will choose the first middle stage switch with the smallest cardinality, that is, middle stage switch 1, and then intersect M_1 with M_2, and M_3, yielding $M_2' = \{1\}$ and $M_3' = \{4\}$. Now, both M_2' and M_3' have a cardinality 1 and we will choose middle

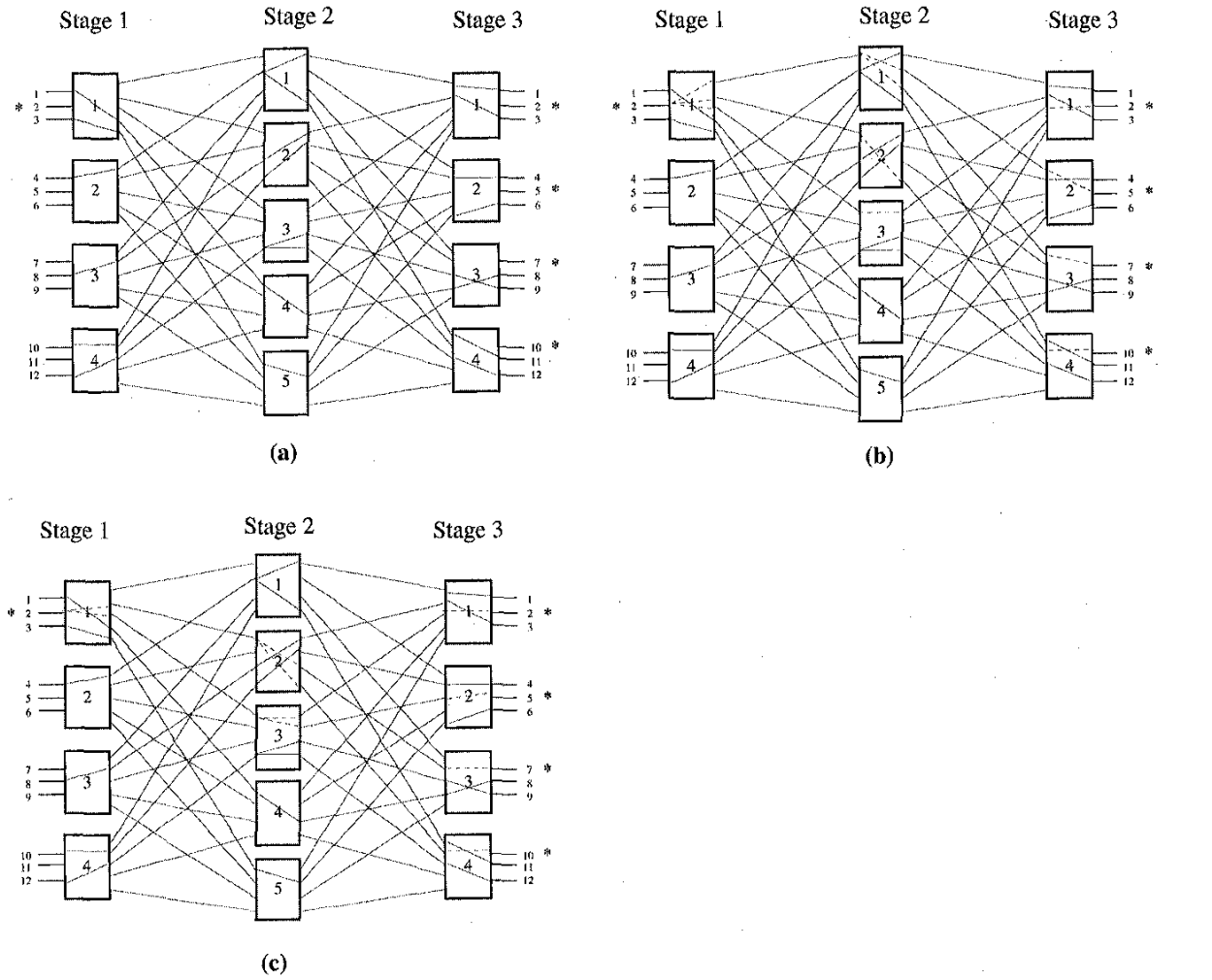

Fig. 3. The difference between Strategy 1 and Strategy 2. (a) The network state before satisfying $I_2 = \{1, 2, 3, 4\}$ in a $v(5, 3, 4)$ network. (b) The network state after satisfying I_2 under Strategy 1. (c) The network state after satisfying I_2 under Strategy 2.

stage switch 2 and intersect M_2' with M_3' yielding $M_3'' = \phi$. Finally, we will choose middle stage switch 3. Thus, we need three middle stage switches for satisfying this connection request. The connections are shown in Fig. 3b. However, under Strategy 2, we will choose the minimum number of available middle switches, and it turns out that only two middle stage switches, middle stage switches 2 and 3, are needed for satisfying this connection request. The connections are shown in Fig. 3c. In general, for a given $v(m, n, r)$ network, it is clear that if we can satisfy all connection requests under Strategy 1, we can definitely satisfy all connection requests under Strategy 2. Of course, since Strategy 2 needs to exam all possible subsets of available middle switches, its complexity is much higher than that of Strategy 1. Even so, it is interesting to see whether Strategy 2 can lead to a tighter nonblocking bound for a $v(m, n, r)$ multicast network.

Strategy 3. *For each input connection request I_i, $i \in \{1, 2, \ldots, nr\}$, use an empty available middle switch (i.e., middle stage switch with no connections) only when there is no subset of nonempty available middle switches that can satisfy the condition in Lemma 1.*

Strategy 3 keeps the total number of nonempty middle stage switches in the network as small as possible at any time, with the intent of reducing the total number of middle stage switches required for nonblocking. Unlike Strategy 2, it achieves some degree of "global optimization" for all input stage switches. We will also analyze the necessary nonblocking condition for the network under Strategy 3.

4 Necessary Nonblocking Conditions for Multicast Networks

In this section, we will derive the necessary conditions under which a $v(m, n, r)$ multicast network is nonblocking in the strict sense (i.e., no control strategy is employed) and under the control strategies outlined in the previous section.

4.1 Combinatorial Properties of Multicast Assignments

In Section 2, we observed that there are at most n 1s, n 2s, ..., n rs distributed in a multicast assignment and, therefore, in the destination sets of middle stage switches in a $v(m,n,r)$ network. In this section, we will discuss some properties of multicast assignments which are useful to the construction of the worst case network states for deriving necessary nonblocking conditions.

The following lemma shows, for integers u, x, and r satisfying some conditions, how to distribute x 1s, x 2s, ..., x rs to u subsets of set $\{1,2,\ldots,r\}$ such that the intersection of any x subsets is nonempty.

Lemma 2. *Given integers u, x, and r, where $u \geq x \geq 1$ and $r = \binom{u}{x}$, there exist u subsets of set $\{1,2,\ldots,r\}$, $S_1, S_2, \ldots, S_u$, such that the flattened set of $\{S_1, S_2, \ldots, S_u\}$ is a multiset chosen from set $\{1,2,\ldots,r\}$ with multiplicity x for each element and the intersection of any x sets of $S_1, S_2, \ldots, S_u$ is nonempty.*

Proof. We construct sets $S_1, S_2, \ldots, S_u$ from x 1s, x 2s, ..., x rs as follows: Initially, let $S_i = \phi$ for $1 \leq i \leq u$. For each x sets $S_{j_1}, S_{j_2}, \ldots, S_{j_x}$, $(1 \leq j_1 < j_2 < \ldots < j_x \leq u)$, chosen from sets $S_1, S_2, \ldots, S_u$, assign a distinct number k, $k \in \{1,2,\ldots,r\}$, to them. This can be done because there are a total of $\binom{u}{x}$ different x sets chosen from $S_1, S_2, \ldots, S_u$, and $r = \binom{u}{x}$. Now, for every x sets $S_{j_1}, S_{j_2}, \ldots, S_{j_x}$ and their corresponding number k, update each set $S_{j_i} (1 \leq i \leq x)$ by adding k to S_{j_i} (i.e., $S_{j_i} \leftarrow S_{j_i} \cup \{k\}$). Note that a total of x ks are added to sets $S_{j_1}, S_{j_2}, \ldots, S_{j_x}$ in this process. The above process is repeated until all x sets chosen from sets $S_1, S_2, \ldots, S_u$ are exhausted. At the end of the construction, we run out of all x 1s, x 2s, ..., x rs. In other words, the flattened set of $\{S_1, S_2, \ldots, S_u\}$ is a multiset chosen from set $\{1,2,\ldots,r\}$ with multiplicity x for each element.

In addition, we can see that the intersection of any x sets of sets $S_1, S_2, \ldots, S_u$ is nonempty. In fact, from the above construction, for any x sets $S_{j_1}, S_{j_2}, \ldots, S_{j_x}$ chosen from sets $S_1, S_2, \ldots, S_u$, we have $\bigcap_{i=1}^{x} S_{j_i} = \{k\}$, where k is the number assigned to these x sets in the above construction. □

Since there are a total of $x \cdot r$ elements used in the construction of sets $S_1, S_2, \ldots, S_u$, we have the following observation:

$$|S_i| = \frac{x \cdot r}{u} = \frac{x\binom{u}{x}}{u} = \binom{u-1}{x-1}, \quad i = 1,2,\ldots,u.$$

For example, let $u = 4$, $x = 2$, and $r = \binom{4}{2} = 6$. According to the above construction, we can distribute two 1s, two 2s, ..., two 6s to sets S_1, S_2, S_3, S_4 as follows:

$$\begin{aligned} S_1 &= \{1, 2, 3\} \\ S_2 &= \{1, 4, 5\} \\ S_3 &= \{2, 4, 6\} \\ S_4 &= \{3, 5, 6\}. \end{aligned}$$

Clearly, the intersection of any two sets of sets S_1, S_2, S_3, S_4 is nonempty, and $|S_i| = \binom{4-1}{2-1} = 3$ for $1 \leq i \leq 4$.

Let's take a look at another example. Let $u = 7$, $x = 3$, and $r = \binom{7}{3} = 35$. We can distribute three 1s, three 2s, ..., three 35s to sets $S_1, S_2, \ldots, S_7$ as follows:

$$\begin{aligned} S_1 &= \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15\} \\ S_2 &= \{1,2,3,4,5,16,17,18,19,20,21,22,23,24,25\} \\ S_3 &= \{1,6,7,8,9,16,17,18,19,26,27,28,29,30,31\} \\ S_4 &= \{2,6,10,11,12,16,20,21,22,26,27,28,32,33,34\} \\ S_5 &= \{3,7,10,13,14,17,20,23,24,26,29,30,32,33,35\} \\ S_6 &= \{4,8,11,13,15,18,21,23,25,27,29,31,32,34,35\} \\ S_7 &= \{5,9,12,14,15,19,22,24,25,28,30,31,33,34,35\}. \end{aligned}$$

It is easy to see that the intersection of any three sets of sets $S_1, S_2, \ldots, S_7$ is nonempty, and $|S_i| = \binom{7-1}{3-1} = 15$ for $1 \leq i \leq 15$.

In general, for any integer n, $x|n$, we can repeatedly apply the above construction technique to distributing n 1s, n 2s, ..., n rs to the subsets of set $\{1,2,\ldots,r\}$ such that the intersection of any x subsets is nonempty. We have the following lemma.

Lemma 3. *For integers u, x, r, and n, where $u \geq x \geq 1$, $r = \binom{u}{x}$, and $x|n$, there exist $v = \frac{n \cdot u}{x}$ subsets of set $\{1,2,\ldots,r\}$, $S_1, S_2, \ldots, S_v$, such that the flattened set of $\{S_1, S_2, \ldots, S_v\}$ is a multiset chosen from set $\{1,2,\ldots,r\}$ with multiplicity n for each element, and the intersection of any x sets of sets $S_1, S_2, \ldots, S_v$ is nonempty.*

Proof. First, the n 1s, n 2s, ..., n rs can be divided into $\frac{n}{x}$ copies of x 1s, x 2s, ..., x rs. By Lemma 2, from one copy of x 1s, x 2s, ..., x rs, we can construct u subsets of set $\{1,2,\ldots,r\}$, say, $S_1, S_2, \ldots, S_u$, such that the intersection of any x sets of them is nonempty. Now, for each of the remaining copies of x 1s, x 2s, ..., x rs, we simply repeat the above construction and obtain another u subsets of set $\{1,2,\ldots,r\}$ equal to $S_1, S_2, \ldots, S_u$, respectively. Thus, using all n 1s, n 2s, ..., n rs, we can construct $v = u \cdot \frac{n}{x}$ subsets of set $\{1,2,\ldots,r\}$, $S_1, S_2, \ldots, S_v$, among which $\frac{n}{x}$ sets are equal to set S_1, $\frac{n}{x}$ sets are equal to set S_2, ..., and $\frac{n}{x}$ sets are equal to set S_u. Clearly, the intersection of any x sets of these v sets is nonempty, and the flattened set of $\{S_1, S_2, \ldots, S_v\}$ is a multiset chosen from set $\{1,2,\ldots,r\}$ with multiplicity $x \cdot \frac{n}{x} = n$ for each element. □

For example, let $u = 4$, $x = 2$, and $r = \binom{4}{2} = 6$. We can distribute n 1s, n 2s, ..., n 6s to sets $S_1, S_2, \ldots, S_v$, where $v = u \cdot \frac{n}{x} = 2n$. In particular, as shown in Fig. 4, the $2n$ sets consist of $\frac{n}{2}$ $\{1,2,3\}$s, $\frac{n}{2}$ $\{1,4,5\}$s, $\frac{n}{2}$ $\{2,4,6\}$s, and $\frac{n}{2}$ $\{3,5,6\}$s. It is easy to see that, for any two sets S_i and S_j $(i \neq j)$, $S_i \cap S_j \neq \phi$.

Now, we have the following fundamental lemma:

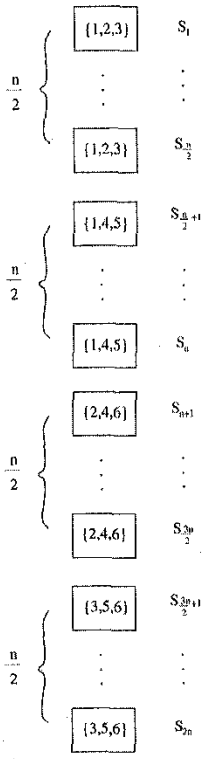

Fig. 4. An example of distributing n 1s, n 2s, ..., n 6s to $2n$ sets.

Lemma 4. *For sufficiently large n, r, and $\frac{m}{n}$ $(m > n)$, there exist $m+n$ subsets of set $\{1,2,\ldots,r\}$,*

$$\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m,$$

which satisfy the following conditions:

1. *The flattened set of*

$$\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$$

is a multiset chosen from set $\{1,2,\ldots,r\}$ with multiplicity of each element no more than n;

2. *For some $x = \Theta\left(\frac{\log r}{\log m - \log n}\right)$ and for any $\mathcal{I}_i$ $(1 \le i \le n)$ and any $\mathcal{M}_{j_1}, \mathcal{M}_{j_2}, \ldots, \mathcal{M}_{j_x}$ $(1 \le j_1 < j_2 < \cdots < j_x \le m)$*

$$\mathcal{I}_i \cap \left(\bigcap_{k=1}^{x} \mathcal{M}_{j_k}\right) \neq \phi.$$

Proof. We give a constructive proof as follows: We will distribute at most n 1s, n 2s, ..., n rs to $n+m$ sets which are subsets of set $\{1,2,\ldots,r\}$, $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$, such that the intersection of any $x+1$ sets of them is nonempty, where x is an integer to be determined later.

Let $t = \frac{m}{n} + 1$, $u = t(x+1)$, and $r' = \binom{t(x+1)}{x+1} \le r$. By Lemma 3, n 1s, n 2s, ..., n r's can be distributed to v subsets of set $\{1,2,\ldots,r'\}$, say, $S_1, S_2, \ldots, S_v$, where $v = t(x+1) \cdot \frac{n}{x+1} = tn = m+n$ such that the intersection of any $x+1$ sets of them is nonempty. We rename these $m+n$ sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$. Thus, the first condition of the lemma holds.

We now show that sets $\mathcal{I}_1$, $\mathcal{I}_2$, ..., $\mathcal{I}_n$, $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_m$ constructed above satisfy the second condition of the lemma. Note that the intersection of any $x+1$ sets of sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ is nonempty, which implies that, for any $\mathcal{I}_i$ $(1 \le i \le n)$ and any $\mathcal{M}_{j_1}, \mathcal{M}_{j_2}, \ldots, \mathcal{M}_{j_x}$ $(1 \le j_1 < j_2 < \cdots < j_x \le m)$

$$\mathcal{I}_i \cap \left(\bigcap_{k=1}^{x} \mathcal{M}_{j_k}\right) \neq \phi.$$

We now determine the value of x. Recall that, in order to apply Lemma 3, we must have $\binom{t(x+1)}{x+1} \le r$. Since we are interested in only the asymptotic bound for x, we can simply replace $x+1$ by x in the above inequality for presentational convenience. Thus, we will derive the bound for x from the following inequality

$$\binom{tx}{x} \le r, \tag{2}$$

where

$$t = \frac{m}{n} + 1 > 2.$$

Note that $\binom{tx}{x} = \frac{(tx)!}{x!(tx-x)!}$, and $y! \sim \sqrt{2\pi y} y^y e^{-y}$ (Sterling's approximation [24]), where e is the natural number and equals $2.718\cdots$.

It can be shown that

$$\binom{tx}{x} \sim \frac{t^x}{\sqrt{2\pi x}\left(1-\frac{1}{t}\right)^{(t-1)x+\frac{1}{2}}}.$$

Therefore, from (2), we have that

$$\frac{t^x}{\sqrt{2\pi x}\left(1-\frac{1}{t}\right)^{(t-1)x+\frac{1}{2}}} \le r.$$

That is,

$$x \le \frac{\log r + \log\sqrt{2\pi x(1-\frac{1}{t})}}{\log t - (t-1)\log(1-\frac{1}{t})} = \frac{\log r + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(1-\frac{1}{t}) + \frac{1}{2}\log x}{\log m - \log n - t\log(1-\frac{1}{t})}.$$

Since $t > 2$, both $|\log(1-\frac{1}{t})|$ and $|t\log(1-\frac{1}{t})|$ are bounded by some constants. In particular, we have

$$\lim_{t\to\infty}\left|t\log\left(1-\frac{1}{t}\right)\right| = \log e.$$

Also, note that $\log x \le \frac{x}{2}$ for $x \ge 2$. Thus, for $0 \le b < 2$ and $x \ge 2$, $x \le a + b\log x$ implies $x \le \frac{a}{1-\frac{b}{2}}$. We can show that, for some constant $c_1 > 0$,

$$x \le c_1 \frac{\log r}{\log m - \log n}. \tag{3}$$

We are actually interested in the maximum x satisfying the condition $\binom{tx}{x} \le r$; that is,

$$\max_x \binom{tx}{x} \le r.$$

Suppose x_0 achieves the maximum in the above inequality. Therefore,

$$\binom{tx_0}{x_0} \le r \le \binom{t(x_0+1)}{x_0+1}.$$

The left inequality implies $x_0 \le c_1 \frac{\log r}{\log m - \log n}$. From the right inequality, we have that

$$\frac{t^{x_0+1}}{\sqrt{2\pi(x_0+1)}(1-\frac{1}{t})^{(t-1)(x_0+1)+\frac{1}{2}}} \ge r.$$

That is,

$$\left[\frac{t}{(1-\frac{1}{t})^{t-1}}\right]^{x_0+1} \ge r\sqrt{2\pi(x_0+1)\left(1-\frac{1}{t}\right)} \ge r.$$

It can be shown that, for some constant $c_2 \ge 0$,

$$x_0 \ge c_2 \frac{\log r}{\log m - \log n}. \tag{4}$$

Thus, from (3) and (4), we have shown that we can choose some $x = \Theta(\frac{r}{\log m - \log n})$ to satisfy the second condition of Lemma 4. □

4.2 Necessary Conditions

In this section, we will construct the worst case network state of a $v(m,n,r)$ multicast network, by applying Lemma 4 such that a certain number of middle stage switches is necessary for achieving nonblocking. Our first theorem reveals the necessary condition for strictly nonblocking multicast networks.

Theorem 1. *The necessary condition for a $v(m,n,r)$ network to be strictly nonblocking for multicast assignments is $m \ge \Theta(n\frac{\log r}{\log\log r})$.*

Proof. We will prove Theorem 1 by contradiction.

Suppose, for a smaller m, say, $m = f(n,r)$, where $f(n,r) = o(n\frac{\log r}{\log\log r})$, the $v(m,n,r)$ network is still nonblocking. Note that the network is strictly nonblocking implies that any state of input stage switches and middle stage switches can be reached. Suppose, at some time, the network reaches a state that the destination sets of middle stage switches M_j becomes $\mathcal{M}_j$, for $j = 1,2,\ldots,m$ and the first input stage switch is empty. Then, there come n input connection requests at the first input stage switch $I_i = \mathcal{I}_i$, $i = 1,2,\ldots,n$, where $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ are the sets described in Lemma 4. By Lemma 4, we know that, for each I_i, the intersection of I_i with the intersection of any x sets from $M_1, M_2, \ldots, M_m$ is not empty. Thus, more than x middle stage switches will be required to satisfy each of input connection requests $I_1, I_2, \ldots, I_n$. Therefore, more than nx middle stage switches in total will be required for satisfying all n input connection requests $I_1, I_2, \ldots, I_n$ on the first input stage switch.

Notice that, by Lemma 4,

$$x = \Theta\left(\frac{\log r}{\log m - \log n}\right).$$

Since $m = f(n,r) = o(n\frac{\log r}{\log\log r})$, for sufficiently large n and r,

$$m < n\frac{\log r}{\log\log r}.$$

Therefore,

$$x > \frac{\log r}{\log\frac{n\log r}{\log\log r} - \log n} = \frac{\log r}{\log\log r - \log\log\log r} \ge c\frac{\log r}{\log\log r}$$

for some constant c. Then,

$$nx > cn\frac{\log r}{\log\log r} > f(n,r) = m.$$

This means that m middle stage switches are not enough to satisfy the input connection requests from the first input stage switch. This contradicts that the network is nonblocking. □

From the proof of Theorem 1, we can obtain a stronger result as follows:

Corollary 1. *The necessary condition for a $v(m,n,r)$ multicast network to be partially strictly nonblocking (i.e., rearrangement is allowed within the same input stage switch) for multicast assignment is $m \ge \Theta(n\frac{\log r}{\log\log r})$.*

Proof. The proof of Theorem 1 holds regardless of the order in which we satisfy the connection requests $I_1, I_2, \ldots, I_n$. □

We now consider the necessary nonblocking conditions under three control strategies described in the previous section.

Theorem 2. *The necessary condition for a $v(m,n,r)$ multicast network to be nonblocking under Strategy 1 is $m \ge \Theta(n\frac{\log r}{\log\log r})$.*

Proof. From Theorem 1, we know that the necessary condition for a $v(m,n,r)$ multicast network to be nonblocking under no control strategy is $m \ge \Theta(n\frac{\log r}{\log\log r})$. To show this is also a necessary condition for Strategy 1, we need only to show that the worse case network state used in the proof of Theorem 1 can be reached under Strategy 1.

Suppose that initially there are no connections in the network. We construct a multicast assignment as follows: First, we generate m input connection requests from m input ports $i_1, i_2, \ldots, i_m$ ($i_j \in \{n+1, n+2, \ldots, nr\}$, $j = 1,2,\ldots,m$) and let

$$I_{i_j} = \mathcal{M}_j, \quad j = 1,2,\ldots,m,$$

where $\mathcal{M}_j$ is defined in Lemma 4. Then, generate n input connection requests from the first input stage switch and

let $I_i = \mathcal{I}_i$, $i = 1, 2, \ldots, n$, where $\mathcal{I}_i$ is given in Lemma 4. Note that I_{i_j}, $j = 1, 2, \ldots, m$, are requested before any input connection requests $I_1, I_2, \ldots, I_n$ in the first input stage switch and the destination sets of all middle stage switches are empty at the beginning. Under Strategy 1, to satisfy I_{i_j}, we will always choose the middle stage switch with smallest cardinality among the available switches until input connection request I_{i_j} is satisfied. Thus, we will end up using one empty middle stage switch for satisfying each of input connection request I_{i_j}, $j = 1, 2, \ldots, m$. After satisfying all I_{i_j}, $1 \le j \le m$, the destination sets of middle stage switches M_j becomes $\mathcal{M}_j$, $j = 1, 2, \ldots, m$, and we still need to satisfy connection requests $I_i = \mathcal{I}_i$, $i = 1, 2, \ldots, n$. Thus, the worst case network state in the proof of Theorem 1 is reached. □

From Theorem 2, we have the following corollary.

Corollary 2. *The necessary and sufficient condition for a $v(m, n, r)$ multicast network to be nonblocking under Strategy 1 is $m \ge \Theta(n \frac{\log r}{\log \log r})$.*

Proof. Combine Theorem 2 with the sufficient condition $m \ge cn \frac{\log r}{\log \log r}$ in [8]. □

Theorem 3. *The necessary and sufficient condition for a $v(m, n, r)$ multicast network to be nonblocking under Strategy 2 is $m \ge \Theta(n \frac{\log r}{\log \log r})$.*

Proof. The proof is similar to that of Theorem 2. That is, we first generate m consecutive connection requests equal to sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ and, then, generate n consecutive connection requests equal to sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$ on an empty input stage switch. Under Strategy 2, we choose the minimum number of middle stage switches for satisfying a connection request. The m connection requests equal to $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ are satisfied first. Note that, when all destination sets of middle stage switches are empty and $\mathcal{M}_i \cap \mathcal{M}_j \neq \phi$ $(i \neq j)$, the minimum number of middle stage switches chosen for satisfying a connection request equal to $\mathcal{M}_j$, $j \in \{1, 2, \ldots, m\}$, is one; that is, we will use one empty middle stage switch to satisfy a connection request. Thus, after satisfying all these m connection requests, the worst case middle stage state in the proof of Theorem 1 can again be reached. In fact, for this particular multicast assignment, there is no difference between Strategy 1 and Strategy 2 in choosing available middle switches for satisfying a connection request.

To prove the sufficiency, note that Strategy 2 chooses the minimum number of middle stage switches for satisfying a connection request, which is smaller than or equal to the number of middle stage switches used under Strategy 1. Thus, the sufficient condition for Strategy 1 is also a sufficient condition for Strategy 2. □

Theorem 4. *The necessary condition for a $v(m, n, r)$ multicast network to be nonblocking under Strategy 3 is $m \ge \Theta(n \frac{\log r}{\log \log r})$.*

Proof. Under Strategy 3, we try to keep the number of nonempty middle stage switches as small as possible at any time. The proof of this theorem is similar to that of Theorem 1. That is, we first assume that we can achieve nonblocking in a $v(m, n, r)$ multicast network under Strategy 3 for some $m = o(n \frac{\log r}{\log \log r})$, which yields $x \ge c \frac{\log r}{\log \log r}$ for some constant c; then we show that this assumption will lead to a contradiction. All we need to do here is to construct a sequence of connection and disconnection requests under Strategy 3 which lead to the worst case network state in which the destination sets of m middle stage switches are equal to sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$, respectively, and the next n connection requests will come from an empty input stage switch and are equal to sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$, respectively, where $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ are specified in Lemma 4.

The following algorithm-like description gives the construction of the sequence of connection and disconnection requests desired.

Initially, all input stage switches and middle stage switches are empty.

Step 1.
Generate n consecutive connection requests equal to sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$, respectively, on the first input stage switch. Under Strategy 3, each of these connection requests is satisfied by occupying an empty middle stage switch as shown in Fig. 5a. Let $m_1 = n$ and $k_1 = 0$. Now, there are a total of m_1 non-empty middle stage switches with destination sets equal to sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$, respectively.

Step i.
$(i \ge 2)$

i.1. Generate k_i $(< n)$ consecutive connection requests equal to sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_i}$ on the ith input stage switch such that these connection requests can be satisfied by exactly m_{i-1} existing nonempty middle stage switches, as shown in Fig. 5b. Sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_i}$ and integer k_i will be described in more detail later. Let

$$m_i = \min\{m_{i-1} + (n - k_i), m\}.$$

i.2. Generate $m_i - m_{i-1}$ consecutive connection requests equal to sets

$$\mathcal{M}_{m_{i-1}+1}, \mathcal{M}_{m_{i-1}+2}, \ldots, \mathcal{M}_{m_i}$$

on the ith input stage switch. Since the first m_{i-1} middle stage switches are used by the earlier connection requests generated in Step i.1 on the ith input stage switch and, therefore, are not available, these $m_i - m_{i-1}$ connection requests can be satisfied only by assigning each connection request an empty middle stage switch as shown in Fig. 5c.

i.3. Generate k_i consecutive disconnection requests on the ith input stage switch which release the connections requested in Step i.1. These disconnection requests are satisfied by

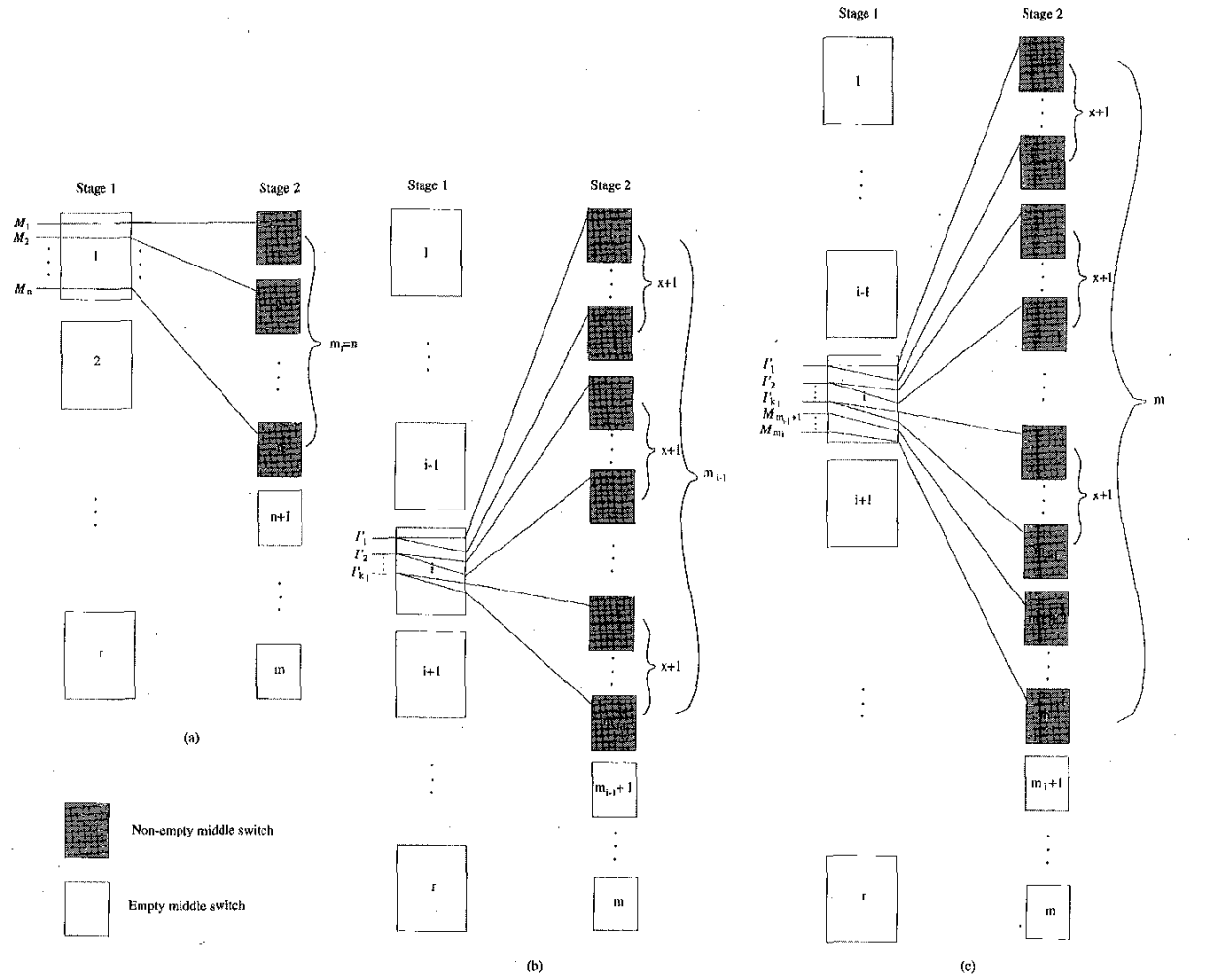

Fig. 5. Illustration of the proof of Theorem 4. (a) Step 1: Each connection request used an empty middle state switch. (b) Step i.1: Each connection request uses $x+1$ nonempty middle stage switches. (c) Step i.2: Each connection request used an empty middle state switch.

releasing the first k_i input ports on the ith input stage switch and their links to middle stage switches. Now, there are a total of m_i nonempty middle stage switches with destination sets equal to sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{m_i}$, respectively.

i.4. If $m_i < m$, go to Step $i+1$, else go to the Last Step.

The Last Step: Generate n consecutive connection requests equal to sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$ on an empty input stage switch.

At this point, we have constructed the desired network state.

Now, we are in the position to describe sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_i}$ and integer k_i in Step i. We know from Lemma 4 that, for any j $(1 \le j \le n)$, a connection request equal to set $\mathcal{I}_j$ can only be satisfied by more than x middle stage switches if the destination sets of available middle switches are equal to some sets among $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$. Therefore, for a specific $y \le x+1$, we can obtain a set $\mathcal{I}'_j \subseteq \mathcal{I}_j$ so that a connection request equal to set $\mathcal{I}'_j$ can be satisfied by exactly y middle stage switches.

For Step 2, let $k_2 = \lceil \frac{m_1}{x+1} \rceil$. We construct $\mathcal{I}'_j \subseteq \mathcal{I}_j$ $(1 \le j \le k_2)$ so that the connection request equal to set $\mathcal{I}'_j$ $(1 \le j \le k_2 - 1)$ can only be satisfied by $x+1$ middle stage switches from m_1 nonempty middle stage switches with destination sets equal to $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{m_1}$ and the connection request equal to set $\mathcal{I}'_{k_2}$ can only be satisfied by the remaining nonempty middle stage switches. Thus, the k_2 connection requests equal to sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_2}$ use up all m_1 nonempty middle stage switches in Step 1.

In general, for Step i $(i > 2)$, let $k_i = \lceil \frac{m_{i-1}}{x+1} \rceil$. By using a similar method as in Step 2, we can construct k_i connection requests equal to sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_i}$ $(\mathcal{I}'_j \subseteq \mathcal{I}_j$ for $1 \le j \le k_i)$ which can use up all m_{i-1} existing nonempty middle stage switches with destination sets equal to $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{m_{i-1}}$ generated in previous $i-1$ steps.

From the construction algorithm, we have following observations.

Observation 1. *The construction of the above sequence of connection and disconnection requests follows Strategy 3.*

In fact, from the construction, we can see that existing nonempty middle stage switches, when available, are always chosen to satisfy the current connection requests in Step i.1 for $i \geq 2$, while only when no nonempty middle stage switches are available for the current connection request, an empty middle stage switch is used to satisfy this connection request in Step 1 and Step i.2 for $i \geq 2$.

Observation 2. *In Step i.1 ($i \geq 2$), we always have $k_i < n$.*

Since $k_i = \left\lceil \frac{m_{i-1}}{x+1} \right\rceil$, if $k_i \geq n$, $m_{i-1} \approx n(x+1) \geq cn\frac{\log r}{\log\log r}$ (where c is a constant) is greater than $m = o(n\frac{\log r}{\log\log r})$ for sufficiently large n and r, which means that the construction already reached the Last Step after Step $(i-1).4$. In other words, the construction never reached Step i.

In addition, since Observation 2 implies $m_i - m_{i-1} \geq 1$, in Step i.2, we use at least one more empty middle stage switch.

Observation 3. *At any time, the elements used in constructing all connection requests are within n 1s, n 2s, ..., n rs.*

Observation 3 follows because, at any time, all connection requests are from sets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ defined in Lemma 4. In fact, in Step i.1 for $i \geq 2$, all connection requests are from sets $\mathcal{I}'_1, \mathcal{I}'_2, \ldots, \mathcal{I}'_{k_i}$, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{m_{i-1}}$; while, in Step i.2 for $i \geq 2$, all connection requests are from sets $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{m_i}$.

Observation 4. *The algorithm-like construction of the sequence of connection and disconnection requests terminates in finite steps (i.e., no infinite loops).*

Since $m_i \leq m$ and, by Observation 2, we have $m_i - m_{i-1} \geq 1$, we conclude that the construction will terminate in no more than m steps.

Observation 5. *There are enough input stage switches in the $v(m,n,r)$ network for the above construction (i.e., the number of input stage switches used in the construction is less than r).*

This observation arises from the following concern: Note that, in each step of the construction, we use one input stage switch. Although, from Observation 4, we can use only a limited number of input stage switches, does this construction run out of all r input stage switches in a $v(m,n,r)$ network?

The proof of Observation 5 is given below.

Suppose the last input stage switch used in the above construction is switch t and note that $k_i = \lceil \frac{m_{i-1}}{x+1} \rceil$. We have that

$$\begin{cases} m_1 = n \\ m_i = m_{i-1} + n - \left\lceil \frac{m_{i-1}}{x+1} \right\rceil, & 1 < i < t \\ m_t = m. \end{cases}$$

Therefore,

$$m_i \geq m_{i-1} + n - 1 - \frac{m_{i-1}}{x+1} = \left(\frac{x}{x+1}\right) m_{i-1} + (n-1).$$

Thus,

$$\begin{aligned} m_i &\geq \left(\frac{x}{x+1}\right)^{i-1} m_1 + \left[\left(\frac{x}{x+1}\right)^{i-2} + \cdots + \left(\frac{x}{x+1}\right) + 1\right] (n-1) \\ &> \frac{1 - \left(\frac{x}{x+1}\right)^i}{1 - \left(\frac{x}{x+1}\right)} (n-1) \\ &= \left[1 - \left(\frac{x}{x+1}\right)^i\right] (x+1)(n-1). \end{aligned}$$

Let $i = t-1$. We have that

$$m = m_t \geq m_{t-1} > \left[1 - \left(\frac{x}{x+1}\right)^{t-1}\right] (x+1)(n-1).$$

That is,

$$\left(\frac{x}{x+1}\right)^{t-1} > 1 - \frac{m}{(x+1)(n-1)}. \tag{5}$$

Note that $\log(1-\alpha) \sim \alpha$ when $\alpha \to 0$. Since

$$x \geq c\frac{\log r}{\log\log r} \quad \text{and} \quad m = o\left(n\frac{\log r}{\log\log r}\right)$$

for sufficiently large n and r, we have that

$$\log\left[1 - \frac{m}{(x+1)(n-1)}\right] \sim \frac{m}{(x+1)(n-1)},$$

$$\text{and} \quad \log\left[1 - \frac{1}{x+1}\right] \sim \frac{1}{x+1}.$$

Hence, from (5), we have that

$$\begin{aligned} t &< \frac{\log\left(1 - \frac{m}{(x+1)(n-1)}\right)}{\log\left(1 - \frac{1}{x+1}\right)} + 1 \sim \frac{\frac{m}{(x+1)(n-1)}}{\frac{1}{x+1}} + 1 \\ &= \frac{m}{n-1} + 1 = o\left(\frac{\log r}{\log\log r}\right). \end{aligned}$$

Clearly, $\frac{\log r}{\log\log r} < r$. Therefore, the number of input stage switches used in the construction can be considered far less than r. That proves Observation 5. □

5 Complexity of Control Strategies and Implementation Issues

In the last section, we obtained the necessary number of middle stage switches required for a multicast network to be nonblocking under different control strategies. We now

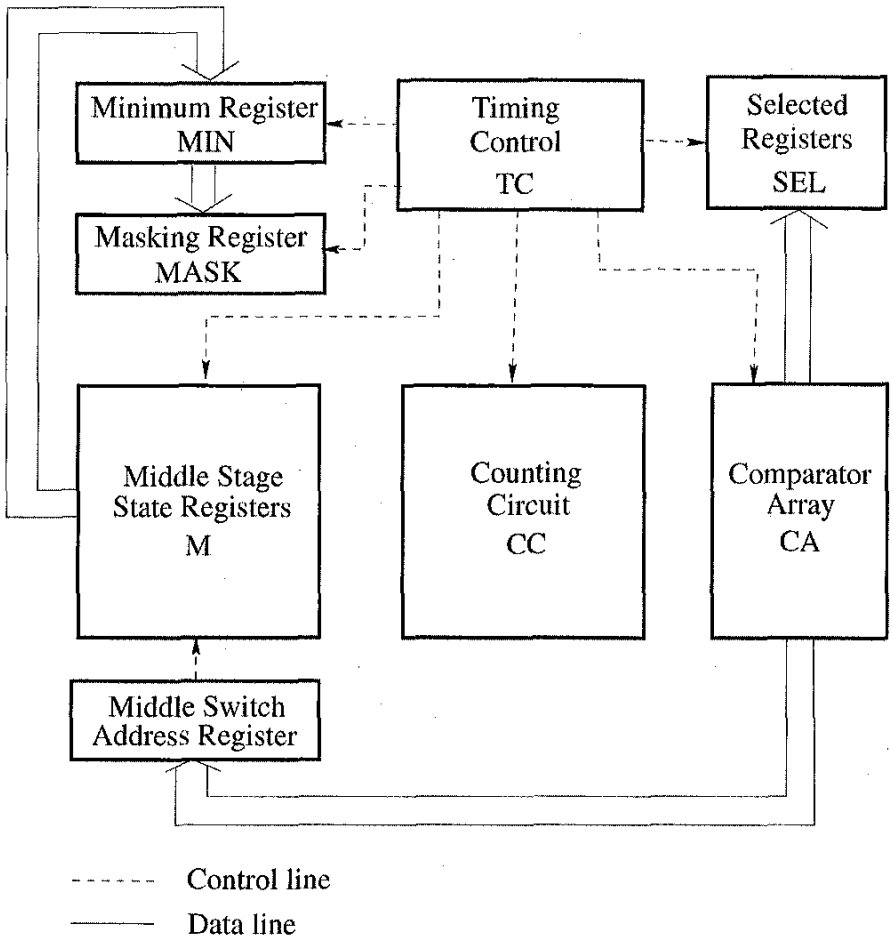

Fig. 6. Overview structure of the hardware network controller.

further compare the complexity of these strategies in order to find a practical control strategy which leads to the minimum network cost and has the lowest time complexity.

From the last section, we know that the necessary and sufficient conditions on the number of middle stage switches required for the network to be nonblocking under both Strategy 1 and Strategy 2 are $m \geq \Theta(n\frac{\log r}{\log\log r})$. Now, let's look at the time complexity of the routing algorithm based on these two control strategies. In [8], it was shown that the routing algorithm based on Strategy 1 has a linear time complexity. However, for Strategy 2, the problem of choosing the minimum number of middle stage switches for satisfying a multicast connection request is NP-complete. In fact, consider the sets of all idle outputs of each available middle switch. The problem of satisfying a multicast connection request under Strategy 2 is equivalent to the problem of finding the minimum number of such sets which cover all destinations of the multicast connection request. Therefore, this problem is exactly the *minimum cover* problem [25], which is NP-complete. This suggests that Strategy 1 is a better strategy than Strategy 2 in terms of time complexity.

Now let's consider the routing algorithm using Strategy 3. It can be implemented by trying all nonempty middle stage switches one by one to see whether the multicast connection request can be collectively satisfied by these nonempty middle stage switches. If not, a single empty middle stage switch will be used for this connection. Thus, the time complexity is still linear. However, it is unknown whether the condition $m \geq \Theta(n\frac{\log r}{\log\log r})$ is sufficient for the network to be nonblocking. Nevertheless, the necessary condition obtained in the last section suggests that Strategy 3 cannot have a lower order sufficient condition than that of Strategy 1.

Furthermore, adopting Strategy 1 has another advantage over other two strategies. Due to the regularity of the algorithm, it can be efficiently implemented in hardware to further reduce the time complexity to $O(\frac{\log^2 r}{\log\log r})$ gate propagations. We briefly discuss this issue next and the detailed implementation can be found in [9].

When considering hardware implementation, we may easily parallelize some frequently performed operations in the control algorithm. For example, we can represent the destination set of a middle stage switch by an r-bit binary

TABLE 1
Summary of the Various Designs of the Network Controller

Design	Gates	Clocks	Gate Delays
Sequential-with-counter	$O(\log r + \log n)$	$O(\frac{nr \log r}{\log \log r})$	–
Sequential-with-adder	$O(r)$	$O(\frac{m \log r}{\log \log r})$	–
Parallel-with-counters	$O(m(\log r + \log n))$	$O(\frac{r \log r}{\log \log r})$	–
Parallel-with-adders	$O(mr)$	$O(\frac{\log r}{\log \log r})$	$O(\frac{\log^2 r}{\log \log r})$

vector with a binary "1" in the ith bit representing the fact that the ith output of the middle stage switch is occupied. Then, the cardinality of a destination set is simply the number of 1s in the vector and can be evaluated either by a sequential circuit, say, a counter, or by a combinational circuit, say, an adder. Furthermore, the evaluation can be performed either sequentially or in parallel among all middle stage switches. Finally, the step of finding the minimum cardinality among all the destination sets can be done by a tree-structure comparator array when the cardinalities are evaluated in parallel. Fig. 6 shows an overview structure of such an implementation, where *middle switch state registers (M)* are used to store the destination set vectors of middle stage switches, the *counting circuit (CC)* is used to count the number of 1s in the destination sets, the *comparator array (CA)* is used to determine the middle stage switch with the minimum cardinality destination set, the *masking register (MASK)* accomplishes the bitwise intersection operations, the *minimum register (MIN)* stores the content of the current minimum cardinality middle switch state register, and, finally, the *selected register set (SEL)* is used to store the middle stage switches selected for satisfying the connection request.

Based on the different hardware cost and the operational time trade-off, four basic types of controller designs were considered in [9]: sequential-with-counter, sequential-with-adder, parallel-with-counters, and parallel-with-adders. We summarize these designs of the network controller in Table 1. From Table 1, we can see that each of hardware controller designs requires only minor additional circuitry beyond that used in the switch modules and that each hardware controller design offers significant improvements in the time required to determine connection path routes when compared with the previous software control algorithm. In particular, in the parallel-with-adders controller design, routing paths for a connection request can be determined in $O(\frac{\log^2 r}{\log \log r})$ gate propagations compared with the $O(nr)$ (or $O(N)$) steps required in the software control algorithm. Such a controller design renders the nonblocking multicast networks useful in the applications which require high-speed network connection path set-ups.

From the above discussion, we can conclude that Strategy 1 is the best strategy among all strategies we have considered and $m \geq \Theta(n \frac{\log r}{\log \log r})$ is optimal for this type of nonblocking multicast network under Strategy 1. As the conclusion for this section, we give several design examples in Table 2 for some practical size nonblocking multicast networks under Strategy 1. The values of r, n, and m shown in the table yield the minimum cost network for the corresponding network size.

TABLE 2
Examples of Optimal $v(m, n, r)$ Nonblocking Multicast Networks under Strategy 1, Where Network Size $N = nr$

Network size N	r	n	m
100	20	5	23
200	20	10	52
400	25	16	89
1000	40	25	155
2000	50	40	260
4000	80	50	343
10000	100	100	710

6 Conclusions

In this paper, we first described several typical routing control strategies for Clos-type nonblocking multicast network. We then derived the necessary conditions under which this type of network is nonblocking for arbitrary multicast assignments in the strict sense as well as under these control strategies. The necessary conditions obtained are represented as the number of middle stage switches $m \geq \Theta(n \frac{\log r}{\log \log r})$. These results match the sufficient nonblocking condition for the currently best available explicitly constructed, constant stage nonblocking multicast network [8], [9]. From these results, we conjecture that the number of middle stage switches for the optimal Clos-type nonblocking multicast network is $O(n \frac{\log r}{\log \log r})$ and that this type of network is optimal for explicitly constructed, constant stage nonblocking multicast network.

ACKNOWLEDGMENTS

Yuanyuan Yang was supported by the U.S. Army Research Office under Grant No. DAAH04-96-1-0234 and by the U.S. National Science Foundation under Grant No. OSR-9350540 and MIP-9522532. A preliminary version of this work appeared in the *Proceedings of the 10th IEEE International Parallel Processing Symposium (IPPS '96)*, Honolulu, Hawaii, pp. 796-802, April 1996.

REFERENCES

[1] L.M. Ni, "Should Scalable Parallel Computers Support Efficient Hardware Multicast?" *Proc. 1995 ICPP Workshop Challenges for Parallel Processing*, pp. 2-7, 1995.
[2] D.K. Panda, "Issues in Designing Efficient and Practical Algorithms for Collective Communication on Wormhole-Routed Systems," *Proc. 1995 ICPP Workshop Challenges for Parallel Processing*, pp. 8-15, 1995.
[3] G.M. Masson and B.W. Jordan, "Generalized Multi-Stage Connection Networks," *Networks*, vol. 2, pp. 191-209, 1972.
[4] 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.
[5] F.K. Hwang and A. Jajszczyk, "On Nonblocking Multiconnection Networks," *IEEE Trans. Comm.*, Vol. 34, pp. 1,038-1,041, 1986.
[6] M. Kumar, "Supporting Broadcast Connections in Benes Networks," IBM Research Report RC-14063, 1988.
[7] P. Feldman, J. Friedman, and N. Pippenger, "Wide-Sense Nonblocking Networks," *SIAM J. Discrete Math.*, vol. 1, no. 2, pp. 158-173, May 1988.
[8] Y. Yang and G.M. Masson, "Nonblocking Broadcast Switching Networks," *IEEE Trans. Computers*, vol. 40, pp. 1,005-1,015, 1991.
[9] Y. Yang and G.M. Masson, "Fast Path Routing Techniques for Nonblocking Broadcast Networks," *Proc. IEEE 13th Int'l Phoenix Conf. Computers and Comm.*, pp. 358-364, Apr. 1994.
[10] Y. Yang, "A Class of Interconnection Networks for Multicasting," *IEEE Trans. Computers*, vol. 47, no. 8, pp. 899-906, Aug. 1998.
[11] Y. Yang and J. Wang, "On Blocking Probability of Multicast Networks," *IEEE Trans. Comm.*, vol. 46, no. 7, pp. 957-968, July 1998.
[12] 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).
[13] C.D. Thompson, "General Connection Networks for Parallel Processor Interconnection," *IEEE Trans. Computers*, vol. 27, pp. 1,119-1,125, 1978.
[14] C.-T. Lea, "A New Broadcast Switching Network," *IEEE Trans. Comm.*, vol. 36, pp. 1,128-1,137, 1988.
[15] C. Lee and A.Y. Oruc, "Design of Efficient and Easily Routable Generalized Connectors," *IEEE Trans. Comm.*, vol. 43, pp. 646-650, 1995.
[16] Y. Yang and G.M. Masson, "Broadcast Ring Sandwich Networks," *IEEE Trans. Computers*, vol. 44, no. 10, pp. 1,169-1,180, Oct. 1995.
[17] X. Lin, P.K. McKinley, and L.M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2D Mesh Multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793-804, Aug. 1994.
[18] P.K. McKinley, H. Xu, A.-H. Esfahanian, and L.M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 12, pp. 1,252-1,265, Dec. 1994.
[19] C. Clos, "A Study of Non-Blocking Switching Networks," *The Bell System Technical J.*, vol. 32, pp. 406-424, 1953.
[20] V.E. Benes, "Heuristic Remarks and Mathematical Problems Regarding the Theory of Switching Systems," *The Bell System Technical J.*, vol. 41, pp. 1,201-1,247, 1962.
[21] A. Itoh et al., "Practical Implementation and Packaging Technologies for a Large-Scale ATM Switching System," *J. Selected Areas in Comm.*, vol. 9, no. 8, pp. 1,280-1,288, 1991.
[22] J. Beetem, M. Denneau, and D. Weingarten, "The GF11 Supercomputer," *Proc. 12th Ann. Int'l Symp. Computer Architecture*, pp. 108-115, 1985.
[23] M.T. Bruggencate and S. Chalasani, "Equivalence between SP2 High-Performance Switches and Three-Stage Clos Networks," *Proc. 25th Int'l Conf. Parallel Processing*, pp. I-1-I-8, 1996.
[24] R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete Mathematics*. Reading, Mass.: Addison-Wesley, 1994.
[25] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman and Company, 1979.

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

Gerald M. Masson received the BSEE degree from the Illinois Institute of Technology, Chicago, in 1966, and the MS and PhD degrees from the Department of Electrical Engineering, Northwestern University, Evanston, Illinois, in 1968 and 1971, respectively. In 1971, Dr. Masson joined the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania. In 1975, he joined the Department of Electrical Engineering, The Johns Hopkins University, Baltimore, Maryland. He spent the 1984-1985 academic year as a visiting professor and half of the 1985-1986 academic year as a professor in the Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania. He returned to Johns Hopkins in 1986, where he is currently a professor and chairman of the Department of Computer Science.

Dr. Masson has served as technical program chairman of numerous IEEE conferences. He served on the editorial boards of *IEEE Transactions on Computers* and the *Journal of Digital Systems* and is currently a member of the editorial board of the *Journal of Electronic Testing*. He is coauthor of the textbook *Distributed Processor Communication Architecture*. Dr. Masson is a fellow of the IEEE and a member of Sigma Xi and Eta Kappa Nu. His current research interests include fault-tolerant computing and computer/communication networks.