A Class of Interconnection Networks for Multicasting*

Yuanyuan Yang
Department of Computer Science and Electrical Engineering
University of Vermont, Burlington, VT 05405
yang@cs.uvm.edu

Abstract

Multicast or one-to-many communications arise frequently in parallel computing applications and in other communication environments. Multicast networks can simultaneously support multiple multicast connections between the network inputs and network outputs. However, due to the much more complex communication patterns and routing control in multicast networks, there is still a considerably large gap in network cost between even 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 multiple multicast connections in a nonblocking fashion and yet have a comparable low cost to permutation networks. We will also provide an efficient routing algorithm for satisfying multicast connection requests in such networks. Moreover, the multicast connecting 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. This enables different system designers to choose the multicast networks which fit in their particular application needs. By utilizing a network with well-defined multicast capability in a parallel computing system, software or algorithm designers of the system will be able to make full use of the multicast capability provided by the network, and substantial improvements in the performance of the system can be achieved due to significantly shortened delays in data transfer and simplified synchronization mechanisms for shared data.

1 Introduction

Multicast or one-to-many connecting capability is highly demanded in parallel applications and in other communication environments. Many parallel applications require that a processor in a parallel computer sends data or messages to some subset of the other processors to complete a common task. One example is the commonly used parallel algorithm for the Fast Fourier Transform (FFT) [1]. This algorithm needs to frequently transpose the data matrix and requires one-to-many communication of a large amount of data among processors. Other examples include Barrier Synchronization [2] and write update/invalidate in directory-based cache coherence protocols [3]. Also, teleconferencing and video broadcasting are typical applications in a telecommunication environment.

There have been growing interests in supporting multicast in parallel computers. Multicast can be supported in either hardware or software. For example, the nCUBE-2 [4] 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 messages are sent at the same time. There has been also much work on supporting multicast in wormhole routed direct networks in software [5, 6]. The basic approach is sending a message along a subset of nodes on the “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 [7, 8] and NEC Cenju-3 [9] 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. Moreover, there has been some work on supporting multicast in software in multistage networks [10, 11].

Since multistage networks can easily have deadlock-free routing and equal communication latency between any network inputs and outputs, they get more and more attention for the interconnecting needs of parallel computers [12] and ATM switch architectures in broadband networks [13]. Also, since multicast is a fundamental communication pattern in parallel computers, efficient hardware support for it becomes increasing important [14]. In this paper, we will be mainly concerned with providing efficient hardware support for multiple multicast communications 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. A multicast network is a network which can realize all possible multicast assignments.

Multicast networks have been extensively studied and there has been much progress in this area [17]–[28]. However, the perceived high network cost and complex routing control of multicast networks might still discourage system designers to seriously consider them for practical parallel computing systems and other communication systems. In fact, due to the much more complex connection patterns in multicast networks, there is still a considerably large gap in network cost between even the currently best known multicast networks and permutation networks. Meanwhile, many real applications may not need full multicast capability. For example, every processor in a parallel computing system may need to have multicast capability to simultaneously send data or control information

---

*This research is supported in part by the National Science Foundation under Grant No. OSR-9350540 and MIP-9522532.
to a group of other processors from time to time, but at any given time, only a small portion of processors may need to perform multicasting, or each processor needs to perform multicasting to only a limited number of other processors. Although permutation networks with multicast switches may realize some multicast connection patterns, they in general can not satisfy the needs of such applications. This is because that permutation networks are designed for realizing only one-to-one connections. In general, there may not be a clear definition on what type of multicast connection patterns a permutation network can realize. Also, the number of multicast connection patterns realized, if any, usually is very limited. This drawback of permutation networks may prevent software and algorithm designers of parallel computing systems from efficiently utilizing multicast capability since there is no simple rule for them to judge whether a given multicast connection can be routed in a single pass through a network and thus no guarantee for the time to complete a multicast connection in the network. In fact, in such a network, a multicast connection may need several passes through the network, depending on the network load and/or state at the time of multicasting, and in the worst case, a multicast connection may have to be performed sequentially.

As discussed above, full multicast networks in general are still too expensive for practical multicast applications, and permutation networks in general cannot support multicast efficiently. Hence, we are motivated to consider compromising network designs for practical multicast applications. In this paper, we will design a class of practical interconnection networks which can realize a substantial amount of well-defined multiple multicast connections and yet have a comparable low 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 proposed networks will enable the software or algorithm designer of a parallel computing system to make full use of the multicast capability provided by the network. By utilizing such a network with well-defined multicast capability, substantial improvements in the performance of a parallel computing system can be achieved due to significantly shortened delays in data transfer and simplified synchronization mechanisms for shared data.

The rest of this paper is organized as follows. Section 2 describes the network structure to be considered. Section 3 gives the necessary definitions and notations for restricted multicast networks. Section 4 reviews the previous results related to this type of networks for both permutation and multicast. Section 5 presents the main results of the paper, the nonblocking conditions for the proposed restricted multicast networks. The routing algorithm is described in Section 6. Finally, Section 7 concludes the paper.

2 The Network Structure

In this section, we provide a brief description of the network we will consider.

The network structure to be considered is a class of networks based on Clos networks[15]. This type of network belongs to so-called constant stage networks or limited stage networks. We know that the network latency of a network is proportional to the number of stages in the network. Therefore, 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)[24]–[28] require at least $\log N$ stages for an $N \times N$ network which represents the minimum network latency that 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[15]. 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 in general sufficient to consider only the three-stage network. A three-stage Clos network with $N_1$ input ports and $N_2$ output ports (i.e. an $N_1 \times N_2$ network) has $r_1$ switch modules of size $n_1 \times m$ in stage 1, $m$ switch modules of size $r_1 \times r_2$ in stage 2, and $r_2$ switch modules of size $m \times n_2$ 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 $v(m,n_1,r_1,n_2,r_2)$ network. In three-stage networks, 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 $v(m,n_1,r_1,n_2,r_2)$ network is shown in Figure 1. For the special symmetrical case where $n_1 = n_2 = n$ and $r_1 = r_2 = r$, the three-stage network is denoted as a $v(m,n,r)$ network. In the following, we will mainly discuss the symmetrical $v(m,n,r)$ networks but the generalization to asymmetrical $v(m,n_1,r_1,n_2,r_2)$ networks is straightforward.

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. From the network structure described above, it is easy to see that the total number of crosspoints of a $v(m,n,r)$ network equals

$$nmr + r^2 m + mnr = m(2nr + r^2) = m(2N + r^2).$$

In other words, the network cost of a $v(m,n,r)$ network is proportional to the number of middle stage switches, $m$ for fixed $N$ and $r$. Therefore, as we will see later, the main focus of the study for this type of networks is on reducing the number of middle switches in such networks to yield lower cost networks.

3 Preliminaries

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

![Figure 1: A general schematic of a $v(m,n_1,r_1,n_2,r_2)$ network.](image-url)
First, it is reasonable to assume that every switch in the $v(m, n, r)$ multicast 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. 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. Moreover, 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 $i$-th 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 $d$ $(1 \leq d \leq r)$ output stage switches at a time (that is, $|I_i| \leq d$), 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 $\alpha (0 \leq \alpha = \alpha(m, n, r) \leq n)$ input connection requests with unrestricted fanouts and all other input connection requests are $d$-restricted $(1 \leq d \leq r)$, we will refer it to an $(\alpha, d)$-multicast assignment. Figure 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 $(\alpha, d)$-multicast assignments as a $v_{\alpha, d}(m, n, r)$ multicast network. Note that in a $v_{\alpha, d}(m, n, r)$ network, those $\alpha$ multicast connections on each input 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 switch does not exceed $\alpha$ at that time. We will simply refer a $v_{\alpha, 1}(m, n, r)$ network, where at most $\alpha$ input ports in each input switch can have unrestricted multicast connections at a time and all other input port can have only one-to-one connections, to a $v_{\alpha}(m, n, r)$ network. Clearly, a $v_{\alpha}(m, n, r)$ 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.

4 Previous Related Work

The $v(m, n, r)$ networks have been extensively studied in the literature[15, 16, 18, 19, 21, 22]. 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 fixed $N$ and $r$. Therefore, the main focus of the study has been on finding the minimum value of the network parameter $m$ for certain type of connecting capability to achieve the minimum network cost.

A recent design[21, 22] shows that a $v(m, n, r)$ network is nonblocking for arbitrary multicast assignments if the number of middle stage switches, $m$, satisfies the condition $m \geq 3(n-1) \frac{\log r}{\log \log r}$. This result represents the currently best known design for constant stage multicast networks. In fact, it has been shown[23] that under several typical routing control strategies $m \geq \Theta (\alpha \frac{\log r}{\log \log r})$ is the necessary condition for a $v(m, n, r)$ multicast network to be nonblocking. However, it was shown[15, 16] that a $v(m, n, r)$ network is nonblocking for permutation assignments if $m \geq 2n - 1$.

Clearly, there is still 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 that $v_{\alpha, d}(m, n, r)$ networks compromise between full multicast networks and permutation networks: they have a comparable low cost to permutation networks and yet powerful enough multicast capability for various multicast applications.

5 Nonblocking Conditions for $v_{\alpha, d}(m, n, r)$ multicast Networks

In this section, we will present the main results of this paper. We will first give the nonblocking condition for general $v_{\alpha, d}(m, n, r)$ multicast networks. We will then extend the result to $v_{\alpha}(m, n, r)$ multicast networks 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 the set of middle stage switches with currently unused links to the input switch associated with input port $i$ the available middle switches. Moreover, for any middle stage switch $j \in \{1, 2, \ldots, m\}$, we will refer the subset of output stage switches to which middle switch $j$ is providing connection paths from the input ports 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 $M_j$'s.

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

**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 $n$ k's in all destination sets $M_1, M_2, \ldots, M_m$. $\square$

![Figure 2: A (2, 1)-multicast assignment in a v(5, 3, 4) network.](image-url)
Now, given a new input connection request $I_i, i \in \{1, 2, \ldots, m\}$, 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 $v_{m,n}(m, n, r)$ network if and only if

$$I_i \cap \bigcap_{k=1}^{x} M_{j_k} = \phi,$$

That is,

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

**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 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 $I_i \cap \bigcap_{k=1}^{x} M_{j_k} = \phi$ before we satisfy this connection request. Otherwise, if exists some output switch $t, t \in I_i \cap \bigcap_{k=1}^{x} M_{j_k}$, then a connection path could not be provided to output switch $t$ through any middle switch in the set of $x$ available middle switches.

□

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

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

$$|I_i \cap M_{j_i}| = \min_{1 \leq j \leq 2n-1} |I_i \cap M_{j_i}|.$$

Then we have

$$|I_i \cap M_{j_i}| \leq \frac{(n-1)f}{2n-1} < \frac{f}{2}.$$

Again, without loss of generality, suppose

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

where $f' < \frac{f}{2}$. Then assign $j_i$ such that

$$|I_i \cap M_{j_i} \cap M_{j_2}| = \min_{1 \leq j \leq 2n-1, j \neq j_i} |I_i \cap M_{j_i} \cap M_{j_2}|.$$

Similarly, we have

$$|I_i \cap M_{j_1} \cap M_{j_2}| < \frac{(n-1)f/2}{2n-1} < \frac{f}{2^2}.$$

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

$$|I_i \cap M_{j_1} \cap M_{j_2} \cdots \cap M_{j_k}| = \min_{1 \leq j \leq 2n-1, j \neq j_1, \ldots, j_{k-1}} |I_i \cap M_{j_1} \cap M_{j_2} \cdots \cap M_{j_k}|.$$

and

$$|I_i \cap M_{j_1} \cap M_{j_2} \cdots \cap M_{j_k}| < \frac{f}{2^k}.$$

There exists some $x \leq \log f$ such that

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

□

**Theorem 2** A $v_{m,n}(m, n, r)$ multicast network is nonblocking if

$$m \geq \alpha(n, r) \log \frac{r}{d} + (n-1)(2 + \log d) + 1.$$

**Proof.*** We will 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 $\alpha(n, r)$ input ports have a fanout $r$ and $(n - 1 - \alpha(n, r))$ 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 $\alpha(n, r) \log r + (n - 1 - \alpha(n, r)) \log d$ available middle switches not available to the new connection request. Again, 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 that after we satisfy $I_i$, we still have $2n - 1 - \log d$ 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 old connection, which yields at least $\log d$ available middle switches. Therefore, in any case, we will 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 $v_{m,n}(m, n, r)$ network is

$$m \geq \alpha \log r + (n - 1 - \alpha(n, r)) \log d + 2n - 1 \geq \alpha \log r + (n - 1)(2 + \log d) + 1.$$

□

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

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

$$m \geq \alpha \log r + (n - 1)(2 + \log d) + 1 \geq \frac{n \beta(r)}{\log r} \log r - \beta(r) + (n - 1)(2 + \beta(r)) + 1.$$
There exists a constant $c$ such that the network is nonblocking if $m \geq cn/\beta(r)$. \hfill \Box

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

Furthermore, we are particularly interested in the restricted multicast networks which have the same order of network cost as the permutation networks. Theorem 4 gives the nonblocking condition for such networks.

**Theorem 4** In a $v_a (m, n, r)$ network, if at most $\frac{cn}{\log r}$, where $c$ is a constant, input ports on each input switch can have unrestricted multicast connections and all other input ports can have one-to-one connections, the nonblocking condition becomes $m \geq (2+c)n-1$.

**Proof.** This condition can be derived by setting $\alpha = \frac{cn}{\log r}$ and $d = 1$ in Theorem 2. \hfill \Box

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 $v_a (m, n, r)$ networks that satisfy the condition in Theorem 4 have the same order of network cost as permutation networks. Theorem 4 suggests that each input switch can have up to $\frac{cn}{\log r}$ input ports out of its $n$ input ports making unrestricted multicast connections at any time while keeping a low 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 multicast, and we can have up to $\frac{cn}{\log r}$ processors in the cluster performing multicast. Moreover, the only thing the higher-level software and algorithm designers need to be concerned is to keep the number of processors doing multicasting in the cluster below the threshold $\frac{cn}{\log r}$. This is a fairly simple rule to judge whether a multicast connection can be realized in a single pass through the network.

Furthermore, we can obtain even lower cost networks for those applications which have weaker requirements for multicast capability. For example, if each input switch has up to $c$ (where $c$ is a constant) input ports making unrestricted multicast connections at any time and all other input ports making only one-to-one connections, the nonblocking condition for a $v_a (m, n, r)$ becomes $m \geq c \log r + 2n - 1$. Also, even if each input switch has up to $c\sqrt{n}$ input ports making unrestricted multicast connections, the number of middle switches needed for nonblocking is only $c\sqrt{n} \log r + 2n - 1$.

We summarize the nonblocking conditions for some typical $v_a (m, n, r)$ 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 amount 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.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nonblocking condition $m$</th>
<th># Unrestricted multicast ports in each input switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permutation</td>
<td>$v(m, n, r)$</td>
<td>$2n - 1$</td>
</tr>
<tr>
<td>$v_a (m, n, r)$</td>
<td>$\alpha = c$</td>
<td>$c \log r + 2n - 1$</td>
</tr>
<tr>
<td>$v_a (m, n, r)$</td>
<td>$\alpha = \frac{0.5n}{\log r}$</td>
<td>$2.5n - 1$</td>
</tr>
<tr>
<td>$v_a (m, n, r)$</td>
<td>$\alpha = \frac{n}{\log r}$</td>
<td>$3n - 1$</td>
</tr>
<tr>
<td>$v_{a,d} (m, n, r)$</td>
<td>$\alpha = \frac{n}{\log r}$</td>
<td>$3n \log \log r$</td>
</tr>
<tr>
<td>$v_{a,d} (m, n, r)$</td>
<td>$d = \log r$</td>
<td>$\frac{n \log \log r}{\log r}$</td>
</tr>
</tbody>
</table>

This enables different system designers to choose the multicast networks which fit in their particular application needs.

### 6 A Routing Algorithm for $v_{a,d} (m, n, r)$ Networks

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

Given a $v_{a,d} (m, n, r)$ network satisfying the nonblocking condition in Theorem 2 and an input connection request $I_i$. Then there are at least $2n - 1$ available middle switches for $I_i$. Take any $k = 2n - 1$ middle switches from them. 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 $j$th input switch. We have following algorithm for connecting $I_i$:
Algorithm:
/* Check the eligibility of the connection request and update \( A[j] \) */
\( j = \left\lceil \frac{\alpha}{2} \right\rceil \);
if \( |I_i| > d \) then {
    if \( (A[j] < \alpha) \) then
    else exit without making connection;
}
/* Find up to \( x = \log |I_i| \) middle switches for \( I_i \) */
\( T M P \leftarrow I_i; \)
\( S \leftarrow \phi; \)
\( T \leftarrow \{1, 2, \ldots, k\}; \)
for \( i = 1 \) to \( x \) do
    \( H[i] \leftarrow \phi; \)
while \( (T M P \neq \phi) \) {
    choose middle switch \( p \) such that
    \( |M_p \cap T M P| = \min_{q \in T} |M_q \cap T M P| \); \( S \leftarrow S \cup \{p\}; \)
    \( T \leftarrow T - \{p\}; \)
    \( H[p] \leftarrow T M P - M_p; \)
    \( T M P \leftarrow M_p \cap T M P; \)
}
/* 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 give some necessary explanations for the routing algorithm. In the above algorithm, set \( S \) 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 Theorem 1 and Theorem 2, we know that at most \( \log |I_i| \) 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.

We now analyze the complexity of the above algorithm. The time for one iteration of the first while loop is proportional to \( |T M P| \cdot |T| \). Since the number of available middle switches is \( 2n - 1 \), after each iteration, \( |T M P| \) reduces its value to half. We know that initially \( |T M P| = |I_i| \leq r \) and \( |T| = 2n - 1 \). Thus, the total time for the first while loop is proportional to \( |I_i| \cdot (2n - 1) \), that is, \( O(N) \). 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. Moreover, by employing the techniques used in[22], we can obtain a parallel routing algorithm for the above routing process with time complexity of \( O(\log^2 r) \).

7 Conclusions

In this paper, we have presented a class of practical interconnection networks for supporting multicast communications in parallel computing systems. The newly designed networks can support a substantial amount of well-defined multicast assignments in a non-blocking 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 connecting 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. By utilizing a network with well-defined multicast capability in a parallel computing system, software or algorithm designers of the system will be able to make full use of the multicast capability provided by the network, and substantial improvements in the performance of the system can be achieved due to significantly shortened delays in data transfer and simplified synchronization mechanisms for shared data.

Acknowledgements

The author would like to thank the anonymous referees for their helpful comments and suggestions.

References


