Performance Evaluation of Switch-Based Wormhole Networks

Lionel M. Ni, *Fellow, IEEE*, Yadong Gui, Sherry Moore

Abstract—Multistage interconnection networks (MINs) are a popular class of switch-based network architectures for constructing scalable parallel computers. Four wormhole MINs built from $k \times k$ switches, where $k = 2^j$ for some j, are considered in this paper: traditional MINs (TMINs), dilated MINs (DMINs), MINs with virtual channels (VMINs), and bidirectional MINs (BMINs). The first three MINs are unidirectional networks, and we show that the cube interconnection pattern can provide contention-free and channel-balanced partitioning of binary cube clusters. BMINs based on butterfly interconnection are essentially a fat tree, and their routing properties are described. Performance comparison among these four networks using simulation experiments is presented with respect to different network traffic patterns. Both DMINs (dilation two) and BMINs have a similar hardware complexity. We conclude that a two-dilated MIN outperforms the corresponding BMIN (or fat tree) for most of the traffic conditions and is a better choice for the design of scalable parallel computers.

Index Terms—Dilated networks, turnaround routing, multistage interconnection networks, wormhole switching, fat tree, scalable parallel computers.

1 Introduction

Switch-based networks, or indirect networks, have emerged or resurged as another promising network architecture for constructing *scalable parallel computers* (SPCs). Most of the switch-based networks are based on some variations of *multistage interconnection networks* (MINs). Such networks can provide bandwidth linear in the number of nodes and latency logarithmic in the number of nodes.

An important metric used to evaluate a network is its communication latency. The *communication latency* equals the elapsed time after the head of a packet has entered the network at the source until the tail of the packet emerges from the network at the destination. The communication latency includes all possible delays encountered during the lifetime of a packet. There are three contentions causing such delays. One is the queuing delay in the source due to many packets to be transmitted by the source, another is link (or buffer) contention due to a busy channel (or buffer full) being using by other packets, and the other is output contention due to the delivery of multiple packets to the same output. These delays reflect the dynamic behavior of the network due to the passing of multiple packets, and may be high if the network traffic is heavy or unevenly distributed.

In addition to the above contentions, the communication latency is highly dependent on the switching technique used [1]. Circuit switching, which has been used in telephone networks, has been adopted in some parallel computers, such as the BBN GP-1000 [2] and TC-2000 [3]. Packet switching, or store-and-forward switching, has been extensively studied for parallel computers (e.g., [4], [5], [6]). In order to offer low communication latency and reduce buffer requirements, wormhole switching [7] has been used in almost all new generation parallel computers. In wormhole switching, each packet[1] is serialized into a sequence of *flits* (flow control digits). The flit at the head of a packet governs the route. As the header flit is routed, the remaining flits follow it in a pipeline fashion. If the header flit(s) encounters a busy channel, the following flits will hold and wait. This property makes wormhole switching susceptible to deadlock. As a result, deadlock avoidance is a critical issue in wormhole-switched networks. An important feature of wormhole switching is that the communication latency is distance-insensitive when there is no channel contention. Because of its low communication latency and the small amount of dedicated buffer space required at each router/switch, wormhole switching has become the most implemented switching technology and will be the only switching method considered in this paper. Interested readers may refer to [1] for a detailed survey of wormhole switching techniques for direct network architectures.

The basic component of those switch-based networks is the small-scale crossbar switches. A $k \times k$ switch has k ports at one side and k ports at the other side of the switch as shown in Fig. 1 with $k = 4$. The value of k is usually 2, 4, 8, or 16, with 4 and 8 as the most popular choices in practical switch-based networks. This paper studies four different designs of switches, and thus four different switch-based networks. Fig. 1a is a traditional unidirectional switch with k input ports and k output ports, where each port has a single unidirectional communication channel. Depending on

• *L.M. Ni is with the Department of Computer Science, Michigan State University, East Lansing, MI 48824-1027; e-mail: ni@cps.msu.edu.*
• *Y. Gui is with the Institute of Computing Technology, Chinese Academy of Science, Wuxi, Jiangsu, P.R. China.*
• *S.Q. Moore is with Sun Microsystems, Mountain View, CA 94043; e-mail: sherry@cps.msu.edu.*

Manuscript received 22 Oct. 1995; revised December 1996.
For information on obtaining reprints of this article, please send e-mail to: transpds@computer.org, and reference IEEECS Log Number D95120.

1. In this paper, the issue of packetization is not considered, and packets and messages are used interchangeably.

the routing control tag, a packet coming from an input port may be forwarded to any of the k output ports. Fig. 1b is a d-dilated ($d = 2$) unidirectional switch, where each port is associated with d unidirectional channels. Depending on the routing algorithm, an outgoing packet may use one of the d channels on a selected output port. With d channels, up to d different packets can be simultaneously transmitted over a port without blocking. Another approach to share the use of a port among multiple packets is to implement virtual channels. As shown in Fig. 1c, two virtual channels share a physical channel in a time multiplexed manner. Each virtual channel has its own flit buffer, control, and data path within the switch. The concept of virtual channels is not new and has been extensively studied for direct networks [8].

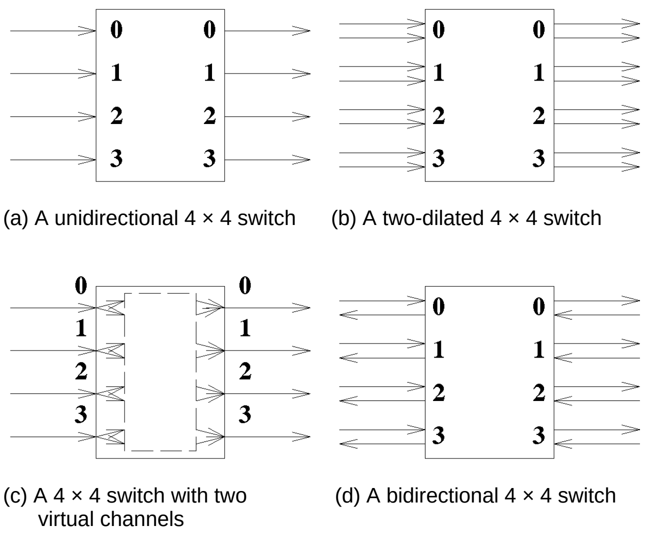

0
1
2
3
(a) A unidirectional 4 × 4 switch
(b) A two-dilated 4 × 4 switch
(c) A 4 × 4 switch with two virtual channels
(d) A bidirectional 4 × 4 switch

Fig. 1. Four different types of 4 × 4 switches.

Unlike the above three unidirectional switches, Fig. 1d illustrates a bidirectional switch in which each port is associated with a pair of opposite unidirectional channels. This implies that two packets can be transmitted simultaneously in opposite directions between neighboring switches. For ease of explanation, it is assumed that processor nodes are on the left-hand side of the network, as shown in Fig. 6. In a $k \times k$ bidirectional switch, a left-hand side port is labeled ℓ_i and a right-hand side port is labeled r_i ($0 \le i \le k - 1$) as shown in Fig. 2. A bidirectional switch supports three types of connections: *forward*, *backward*, and *turnaround* (see Fig. 2). For simplicity, the input device on port ℓ_i (r_j) is denoted as *input port* ℓ_i (r_j), and the output device on port ℓ_i (r_j) is denoted as *output port* ℓ_i (r_j). In forward connection, input port ℓ_i is connected to output port r_j, where $0 \le i, j \le k - 1$. In backward connection, input port r_i is connected to output port ℓ_j, where $0 \le i, j \le k - 1$. In turnaround connection, input port ℓ_i is connected to output port ℓ_j, where $0 \le i \ne j \le k - 1$. No connection is allowed from input port r_i to output r_j, where $0 \le i \ne j \le k - 1$. This property prevents the shortest-path routing from deadlock.

Note that in this paper a channel refers to a unidirectional communication channel. Strictly speaking, the switches in Fig. 1b and Fig. 1d should be considered as $2k \times 2k$ switches. However, in order to make our explanation easier, we will

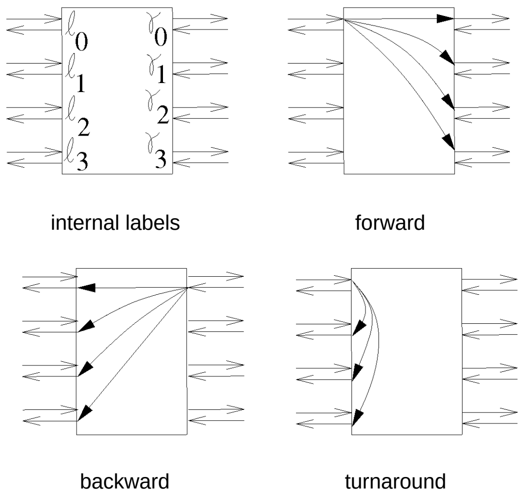
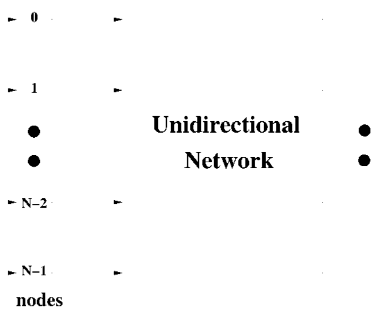


Fig. 2. Possible connection patterns within a 4 × 4 bidirectional switch.

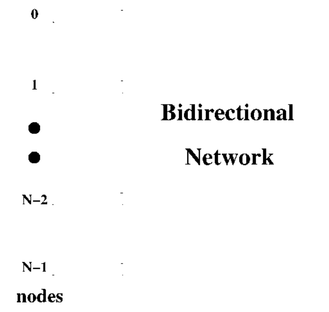
refer all those four types of switches in Fig. 1 as 4×4 switches with the understanding that some ports may have multiple communication channels.

Scalable parallel computers, typically based on distributed-memory architecture, are composed of a number of nodes, where each node has its own processor(s), local memory, and other supporting devices. The primary function of the network in an SPC is to route packets among those interconnected nodes. In this paper, four different switch-based networks based on those four different types of switches are considered as the network architecture for SPCs. A generic SPC based on those unidirectional switches is shown in Fig. 3a, whereas Fig. 3b shows a generic SPC based on bidirectional switches. Processor nodes represented by circles are interconnected through the switch-based network. In this paper, it is assumed that there is exactly one pair of input channel and output channel connecting a node to the network, resulting in so-called "one-port communication architecture." This assumption, which is consistent with many existing parallel computers, implies that the local processor must transmit (receive) packets in sequence. In order to make a fair comparison, we also assume that the input and output channel bandwidths are the same for all nodes in all network architectures.

The main objective of this paper is to evaluate and compare the performance of those four different MINs based on wormhole switching. Our main performance metric is communication latency. There are some other issues or metrics, such as packaging and scalability, related to interconnection networks [9]. However, these issues don't make any significant difference among these MINs studied in this paper and will not be considered. Section 2 describes those unidirectional MINs using the three different unidirectional switches. Two known MIN topologies, butterfly MIN and cube MIN, will be considered. Section 3 discusses bidirectional MINs. Properties and routing algorithms of the butterfly bidirectional MIN will be detailed. Section 4 will address the network partitionability and traffic localization issues. Performance evaluation and comparison based on simulation experiments is reported in Section 5. Section 6 concludes the paper and indicates future work.



(a) unidirectional network



(b) bidirectional network

Fig. 3. Two generic switch-based scalable parallel computer architectures.

2 Unidirectional MINs

An N-node MIN built with $k \times k$ switches can be represented as

$$C_0(N)G_0(N/k)C_1(N) \ldots C_{n-1}(N)G_{n-1}(N/k)C_n(N)$$

where G_i refers to the ith stage, C_i refers to the ith connection, and $N = k^n$. There are n stages. Each stage G_i consists of N/k identical $k \times k$ switches and thus is denoted as $G_i(N/k)$. Each connection C_i connects N right-hand side ports at stage G_{i-1} to N left-hand side ports at stage G_i and thus is denoted as $C_i(N)$. A connection pattern C_i defines the topology of the one-to-one correspondence between two adjacent stages, G_{i-1} and G_i, also known as a *permutation*.

There are many ways to interconnect adjacent stages. Banyan networks are a class of MINs with the property that there is a unique path between any pair of source and destination [10]. An N-node ($N = k^n$) Delta network is a subclass of banyan networks, which is constructed from identical $k \times k$ switches in n stages, where each stage contains (N/k) switches. A unique property of Delta networks is their self-routing property [11]. Many of the known MINs, such as Omega, flip, cube, butterfly, and baseline, belong to the class of Delta networks [11] and have been shown to be topologically and functionally equivalent [12]. A good survey of those MINs can be found in [13].

This paper considers two popular interconnection patterns between adjacent stages: butterfly and perfect shuffle, which are formally defined below.

Definition 1. *The* ith *k-ary butterfly permutation* β_i^k, *for* $0 \le i \le n-1$, *is defined by*

$$\beta_i^k(x_{n-1} \ldots x_{i+1}x_ix_{i-1} \ldots x_1x_0) = x_{n-1} \ldots x_{i+1}x_0x_{i-1} \ldots x_1x_i$$

where $0 \le x_i \le k-1$.

Definition 2. *The perfect k-shuffle connection* σ *is defined by*

$$\sigma(x_{n-1}x_{n-2} \ldots x_1x_0) = x_{n-2} \ldots x_1x_0x_{n-1}$$

where $0 \le x_i \le k-1$.

Two topologically equivalent MINs are considered below. Both MINs are a class of Delta networks. The self-routing property of these two MINs allows the routing decision to be determined by the destination address. For a $k \times k$ switch, there are k output ports. If the value of the corresponding routing tag is i ($0 \le i \le k-1$), the corresponding packet will be forwarded via port i. For an n-stage MIN, the routing tag is $T = t_0t_1 \ldots t_{n-1}$, where t_i controls the switch at stage G_i.

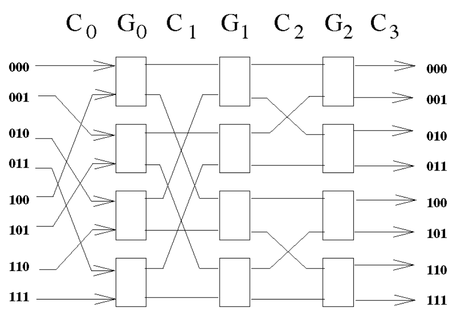
Butterfly MINs. In a butterfly MIN, connection pattern C_i is described by the ith butterfly permutation β_i^k. As indicated in Definition 1, the ith butterfly permutation interchanges the 0th digit and the ith digit of the index. β_0^k is selected to be connection pattern C_n. For a given destination $d_{n-1}d_{n-2} \ldots d_0$, the routing tag is formed by having $t_i = d_{i+1}$ for $0 \le i \le n-2$ and $t_{n-1} = d_0$.

Cube MINs. In a cube MIN (also known as indirect cube or multistage cube [13]), connection pattern C_i is described by the $(n-i)$th butterfly permutation β_{n-i}^k for $1 \le i \le n$. C_0 is selected to be σ. For a given destination $d_{n-1}d_{n-2} \ldots d_0$, the routing tag is formed by having $t_i = d_{n-i-1}$ for $0 \le i \le n-1$.

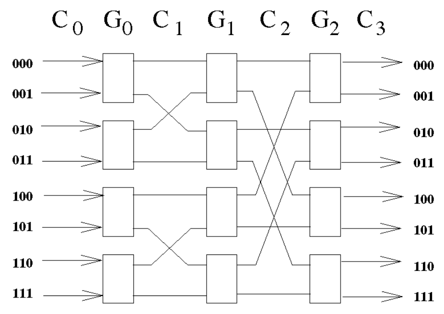
Fig. 4 illustrates two such eight-node MINs. We shall refer to these traditional MINs as TMINs. Such TMINs have been extensively studied in the past and have been adopted in many research prototype parallel computers, such as the Illinois Cedar [14], the Purdue PASM [15], the IBM RP3 [16], and the NYU Ultracomputer [17]. Some commercial parallel computers have also adopted such networks, such as the BBN GP-1000 ($k = 4$) [2], TC-2000 ($k = 8$) [3], Monarch ($k = 8$) [18], and the NEC Cenju-3 ($k = 4$) [19]. Note that the perfect shuffle connection before stage G_sub0 makes the main difference between butterfly MINs and cube MINs. As will be shown later, this perfect shuffle connection plays an important role to the network partitionability. Both the BBN butterfly (GP-1000 and TC-2000) and the NEC Cenju-3 only support straight connection on input and output sides. The GP-1000 and TC-2000 use circuit switching.[2] The NEC Cenju-3 adopts wormhole switching.

An important characteristic which distinguishs between these two TMINs is their ability to be partitioned into different clusters without contention. This issue will be discussed in Section 4. It is known that both cube TMINs and butterfly TMINs are topologically and functionally equivalent [12]. However, we will show in Section 4 that the cube

2. With circuit switching, reply messages, such as acknowledgment, can be sent back via the same path.



(a) an eight-node cube TMIN



(b) an eight-node butterfly TMIN

Fig. 4. Two eight-node TMINs built with 2 × 2 switches.

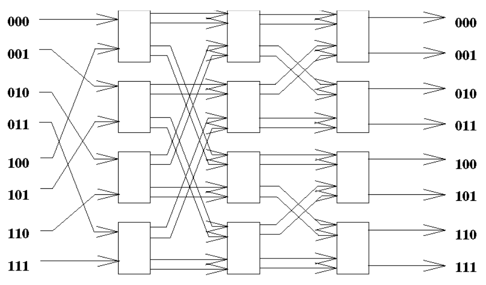
TMIN can more evenly utilize the communication channels than the butterfly TMIN when the network is partitioned and the network is unidirectional.

2.1 Dilated MINs (DMINs)

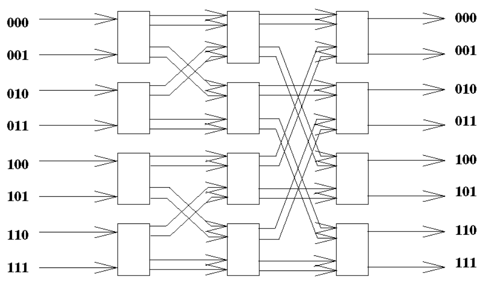
One of the nice features of the above TMINs is that there is a simple algorithm for finding a path of length log_kN between any input and output pair. However, if a link becomes congested or fails, the unique path property can easily disrupt the communication between some input and output pairs. The congestion of packets over some channels causes the known *hot spot* problem [20]. Many solutions have been proposed to resolve the hot spot problem. A popular approach is to provide multiple routing paths between any source and destination pair so as to reduce network congestion as well as to achieve fault tolerance. These methods usually require additional hardware, such as extra stages or additional channels.

For ease of comparison purpose, the dilated MINs are considered in this paper [5]. In a *d*-dilated MIN (DMIN), each switch is replaced by a *d*-dilated switch (see Fig. 1b). By using replicated channels, DMINs offer substantial network throughput improvement [5]. Design of dilated networks has received much attention recently (e.g., [21]). Fig. 5 shows a two-dilated cube MIN and a two-dilated butterfly MIN with eight nodes. Note that half of the input channels and half of the output channels to/from the network are not used in order to maintain the one-port communication architecture and to make a fair comparison with bidirectional MINs.

The routing tag of a DMIN can be determined by the destination address as mentioned for TMINs. Within the network switches, packets destined for a particular output port are randomly distributed to one of the free channels of that port. If all channels are busy, the packet is blocked. The in-



(a) a cube DMIN (dilation two)



(b) a butterfly DMIN (dilation two)

Fig. 5. Two two-dilated MINs using two-dilated 2 × 2 switches.

crease of throughput in dilated networks is obtained by adding redundancy to the network. Note that the bottleneck of a dilated network may be caused due to output contention.

2.2 MINs with Virtual Channels (VMINs)

It is quite expensive to replicate each channel in a wormhole-switched network with its own unique set of physical wires. Furthermore, in most applications, the channel utilization is not high. A virtual channel is a logical channel with its own flit buffer, control, and data path [8]. A virtual channel may share a physical communication channel with other virtual channels. An important issue concerning virtual channels is the multiplexing of a physical channel among many virtual channels. The multiplexing technique should be designed to maximize the channel utilization. Specifically, if m virtual channels share a physical channel with bandwidth W, and k virtual channels are active, where $1 \le k \le m$, then each active virtual channel should have an effective bandwidth of W/k. Since the number of active virtual channels is a function of time, the switch should be able to dynamically allocate channel bandwidth to the active virtual channels. To guarantee fairness, channel multiplexing is usually accomplished at the flit level.

The virtual channel concept does have a drawback. Consider the following scenario in which a communication path traverses multiple physical channels, each of which supports many virtual channels. If the bandwidth of each physical channel is W and there is no sharing with other virtual channels, the effective bandwidth of the communication path is W. On the other hand, if one of the physical channels is shared with three other packets, then the effective bandwidth of the entire path is reduced to $W/4$, even though the available bandwidth of the other channels in the path is W. Another drawback is the increased flit processing delay within each switch, and thus long cycles [22]. Although the concept of virtual channels has been implemented in some direct networks, such as the Cray T3D [23] and the MIT Reliable Router [24], to the best of our knowledge, this concept has not yet being implemented in any switch-based wormhole networks. Performance comparison between virtual channel MINs (VMINs) and other MINs will be studied in Section 5.

3 Bidirectional MINs

To allow for bidirectional communication, each port of the switch has dual channels as shown in Fig. 1d. For ease of explanation, it is assumed that processor nodes are on the left-hand side of the network. The system architecture of an eight-node butterfly bidirectional MIN (BMIN) is illustrated in Fig. 6. Note that due to turnaround connection, the cube interconnection is apparently not a good choice, which will become more clear in Section 3.3. Available ports on the right-hand side of the network are used to configure larger networks, which are not shown in the figure.

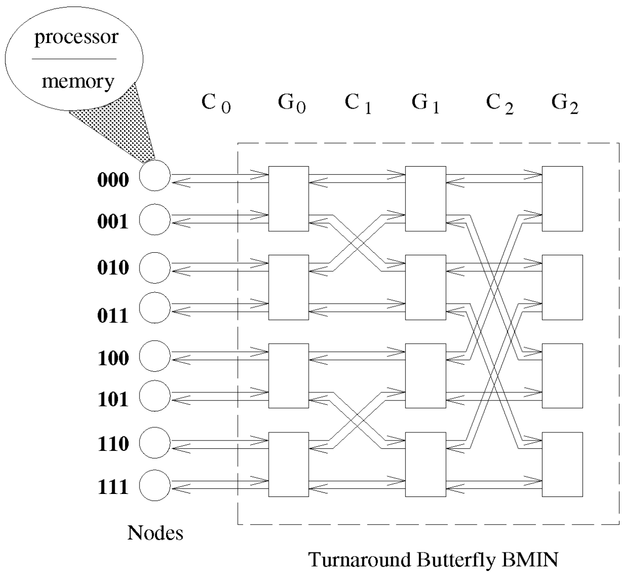


Fig. 6. An eight-node bidirectional butterfly MIN.

The concept of bidirectional switches was studied in [25]. There are many commercial SPCs using BMINs with wormhole switching and turnaround routing including the TMC CM-5 [26], Meiko CS-2 ($k = 4$) [27], and IBM SP-1/2 ($k = 4$) [28], [29]. In the CM-5, the first two level stages use 4×2 switches, yielding a dual-port communication architecture. Although BMINs have been used in many commercial machines, to the best of our knowledge, the routing property of BMINs has not been formally described.

3.1 The Turnaround Routing

In a butterfly BMIN built with $k \times k$ switches, source address S and destination address D are represented by k-ary numbers $s_{n-1} \ldots s_1 s_0$ and $d_{n-1} \ldots d_1 d_0$, respectively. The function *FirstDifference*(S, D) returns i, the position where the first (leftmost) different digit appears between $s_{n-1} \ldots s_1 s_0$ and $d_{n-1} \ldots d_1 d_0$. More formally, it can be defined as follows.

Definition 3. *FirstDifference(S, D) = t if and only if $s_t \neq d_t$ and $s_j = d_j$ for $t < j < n$.*

A turnaround routing path between any source and destination pair is formally defined below.

Definition 4. *A turnaround path is a route from a source node to a destination node. The path must meet the following conditions:*

- *the path consists of some forward channel(s), some backward channel(s), and exactly one turnaround connection;*
- *the number of forward channels is equal to the number of backward channels; and*
- *no forward and backward channels along the path are the channel pair of the same port.*

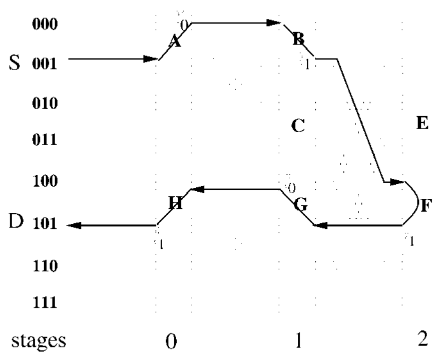
Note that the last condition is to prevent redundant communication from occurring. Also if shortest-path routing is required, this condition is not necessary.

To route a message from source to destination, the message is first sent forward to stage G_t. It does not matter which switch (at stage G_t) the message reaches. Then, the message is turned around and sent backward to the destination. As it moves forward to stage G_t, a message may have multiple choices as to which forward output channel to take. The decision can be resolved by randomly selecting from among those forward output channels which are not blocked by other messages. After the message has attained a switch at stage G_t, it takes the unique path from that switch backward to its destination. The backward routing path can be determined by the "destination tag" routing: the message takes output channel ℓ_{dj} on a switch at stage G_j. The turnaround routing algorithm is shown in Fig. 7. Note that this is a distributed routing algorithm, in which each switch determines the output channel based on the address information carried in the message.

```
Algorithm: Turnaround routing in each switch at stage $j$
Input: Source address $S$: $s_{n-1} \ldots s_1 s_0$
    Destination address $D$: $d_{n-1} \ldots d_1 d_0$
Procedure:
    1. $t$: = $FirstDifference(S, D)$ (* $j \le t$ is always true. *)
    2. If $j = t$, then take a turnaround connection to port $\ell_{d_j}$.
    3. If $j < t$ and the message comes from an input port
        $\ell_m$, then take a forward connection to any avail-
        able port $r_i (0 \le m, i \le k-1)$.
    4. If $j < t$ and the message comes from an input port
        $r_m$, then take a backward connection to port
        $\ell_{d_j} (0 \le m \le k-1)$.
```

Fig. 7. The turnaround routing algorithm executed in each switch at stage j.

Fig. 8 shows an eight-port butterfly BMIN built with 2×2 switches. In Fig. 8, the function *FirstDifference*(001, 101) returns 2. The message is first sent from S to any switch at stage G_2, say F. Note that path $A \to B \to F$ is randomly selected. An alternative could be $A \to C \to E$. Then, the message is turned around in F and sent backward to D. Path $F \to G \to H \to D$ is the unique path from F to D. This path is determined by taking output channel ℓ_1 on F, output channel ℓ_0 on G, and output channel ℓ_1 on H.



FirstDifference(S, D) = 2

Fig. 8. An example of turnaround routing.

3.2 Properties of the Turnaround Routing

Some properties of the turnaround routing in butterfly BMINs are described in this section.

3.2.1 Deadlock-Free Routing

A critical issue in the design of routing algorithms based on wormhole switching is to avoid deadlock [1]. Since a message only turns around once from a forward channel to a backward channel, the dependency graph for the routing paths selected in this way is free from cycles. Therefore, the turnaround routing is deadlock free.

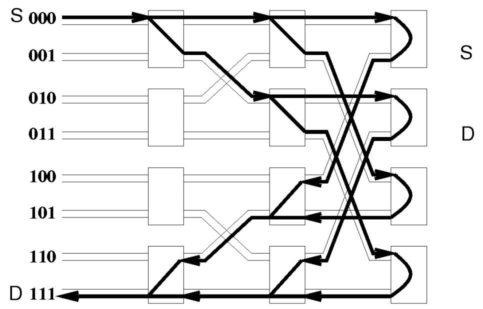
3.2.2 Number of Shortest Paths

By the butterfly connection, any path connecting source S and destination D must pass through a switch at stage G_t where t = *FirstDifference*(S, D). Therefore, the turnaround routing is the shortest-path routing. On the other hand, however, when it moves forward, a message can choose an arbitrary forward channel at a switch (there is no redundancy for backward channels). There are multiple choices of the shortest path which the turnaround routing may select between a source and a destination. This property can be formalized as follows:

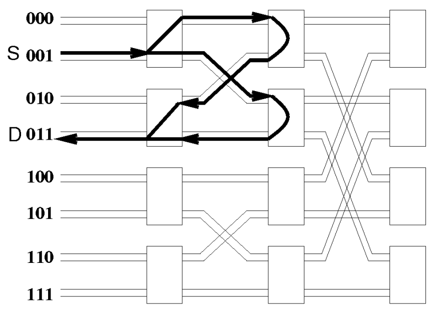
THEOREM 1. *In an N-node butterfly BMIN built with* $k \times k$ *switches* ($N = k^n$), *there are* k^t *shortest paths between source S and destination D, each of which can be generated by the turnaround routing, where t = First Difference(S, D).*

Fig. 9 shows two examples in an eight-node butterfly BMIN with 2×2 switches. Fig. 10 shows two other examples in a 16-node butterfly BMIN with 4×4 switches.

Although there may exist multiple paths for a given source and destination pair, the butterfly BMIN with turnaround routing is a blocking network like unidirectional MINs. In a blocking network, any source node may not be connected to any destination node without affecting the existing connections. For example, in Fig. 11, the message sent from node 011 to node 111 and the message sent from node 001 to node 110 contend for a common channel, output channel ℓ_1 at the bottom switch in stage G_2. It is assumed that a source node has no knowledge of the traffic in the network. Under the circumstance that some (backward) channels are obstructed by other traffic, a message cannot predict a “correct” routing path to avoid channel collision. As a result, a “wrong” one might be chosen such that the message contends for backward channels with other messages.

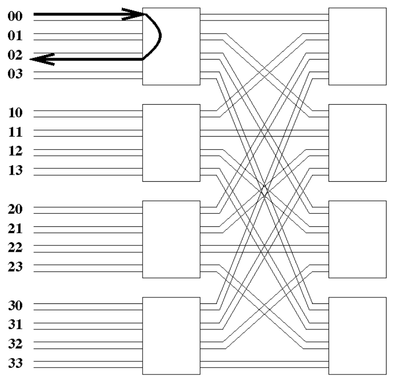


(a) FirstDifference(S, D) = 2 and four shortest paths

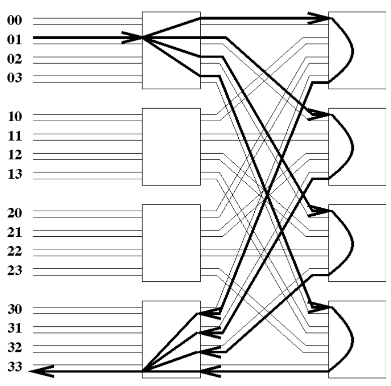


(b) FirstDifference(S, D) = 1 and two shortest paths

Fig. 9. Multiple shortest paths with 2 × 2 switches.



(a) one path



(b) four paths

Fig. 10. Multiple shortest paths with 4 × 4 switches.

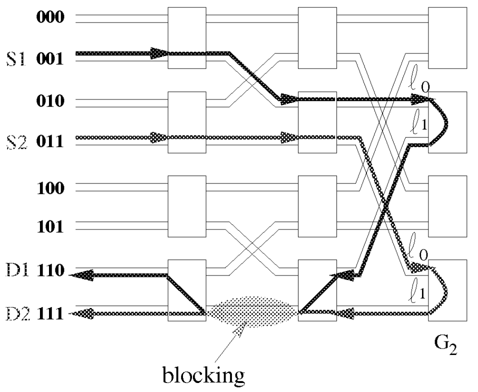


Fig. 11. Blocking network.

3.2.3 Path Length

The path length is defined as the number of channels that a packet has to traverse in the network. For unidirectional MINs, the path length is a constant $n + 1$. For BMINs, the path length is dependent on the location where the packet makes a turn, which is $2(t + 1)$, where $t = \mathit{FirstDifference}(S, D)$.

By taking a closer look, the right-most stage is redundant and can be eliminated when $k = 2$, as shown in Fig. 12. For larger switches, the right-most stage can also be removed at the expense of reduced number of routing paths. For ease of discussion, we assume the existence of the right-most stage.

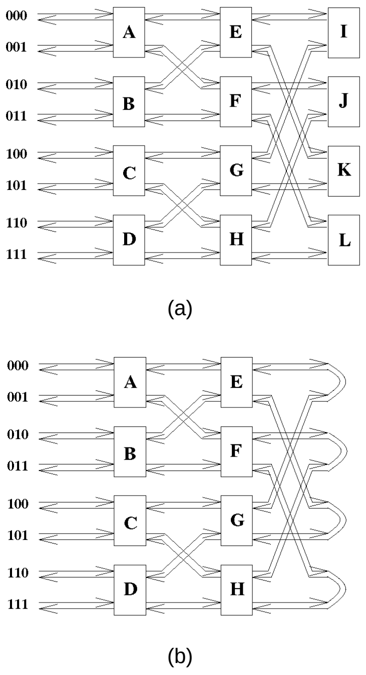


Fig. 12. Topological equivalence by removing the right-most stage.

3.3 Analogy to Fat Tree

As shown in Fig. 13, a butterfly BMIN with turnaround routing can be viewed as a fat tree [30]. In a fat tree, processors are located at leaves, and internal vertices are switches. When a message is routed from one processor to another, it is sent up (in forward direction) the tree to the least common ancestor of the two processors, and then sent down (in backward direction) to the destination. Such a tree routing also explains the turnaround routing.

A subtree rooted at an interior vertex represents a subnetwork partition in a multistage network. For example, subnetworks “A,” “B,” and “C” in Fig. 13a corresponds to subtrees rooted at interior vertices “A,” “B,” and “C” in Fig. 13b, respectively. The communication locality in a subnetwork can be easily explained by that in a subtree. Furthermore, in the corresponding fat tree (Fig. 13b), the number of outgoing parent connections from an interior vertex is equal to the number of leaves in the subtree rooted at that interior vertex. When it is sent up, a message can take any available outgoing channel connecting to the parent vertex. However, each incoming (backward) channel constituting the path from an interior vertex to a leaf is unique.

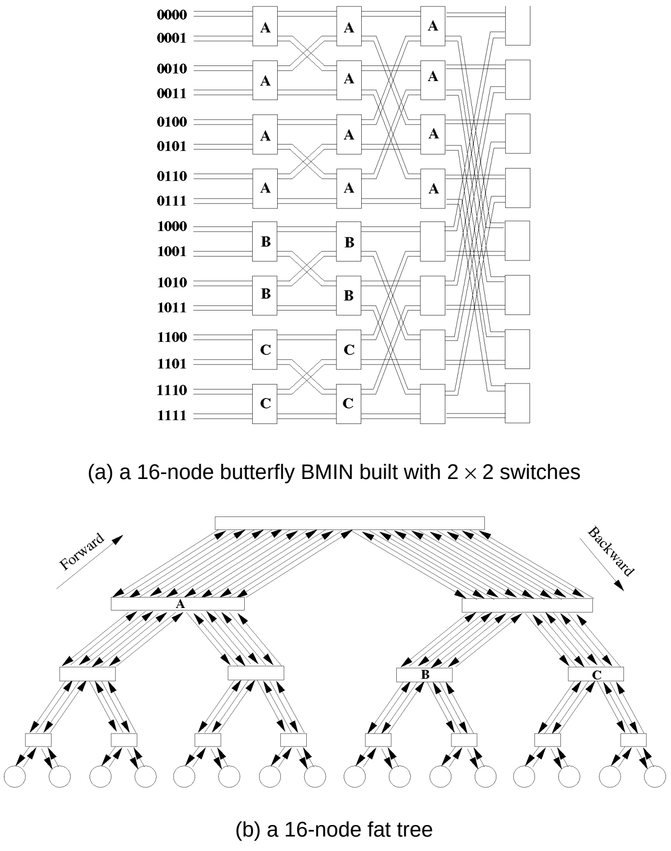


Fig. 13. Fat tree and butterfly BMIN.

4 Network Partitionability and Traffic Localization

In a typical scalable parallel computer, the processors are allocated to different jobs, where each job (or application) usually has an exclusive subset of processors, called a *processor cluster*. If the system does not support contention-free processor allocation, network communication interference among processor clusters will affect the application performance. This section discusses the network partitionability and traffic localization issues. In this paper, we consider those processor clusters forming a cube defined below.

DEFINITION 5. *In a MIN with $N = k^n$ nodes, a k-ary m-cube (cube) consists of k^m nodes which have the same $n - m$ radix-k digits (fixed variables) in their node addresses, where these same digits can be in any $n - m$ locations of the n possible locations. Two cubes are disjoint if they have different fixed variables and one is not a subset of the other.*

DEFINITION 6. *A k-ary m-cube is referred to as a base k-ary m-cube (base cube) if these $n - m$ digits are in the most significant $n - m$ locations (or the remaining m digits are in the least significant m locations) of their node addresses.*

A base cube is a special case of a cube. Consider a system with $N = 4^4$ nodes. The cluster (21**) has 16 nodes ranging from (2100) to (2133) and is a base four-ary two-cube. The cluster (3*1*) has 16 nodes ranging from (3010) to (3313) and is a four-ary two-cube.

In addition to guaranteeing contention-free[3] network partitioning, it is important that the number of communication channels between two adjacent stages is the same as the number of nodes in the corresponding cluster. Thus, if a cluster has c nodes, the number of channels (or channel pairs) allocated to the cluster should be c between any two adjacent stages, and this is referred to as *channel-balanced allocation*. If the number of channels allocated is less than c, it implies the possibility of channel congestion within that cluster. If the number of channels allocated is greater than c, it implies that other clusters may be allocated with less number of channels or the channels have to be shared with other clusters.

LEMMA 1. *A cube unidirectional MIN with $N = k^n$ nodes can be partitioned into contention-free and channel-balanced disjoint k-ary cubes.*

PROOF. We shall prove that the number of channels used by an m-cube cluster is always k^m between any two adjacent stages and the channels from different clusters (k-ary cubes) are always distinct.

Consider a source and destination pair: $s_{n-1} \cdots s_0$ and $d_{n-1} \cdots d_0$. The channels before entering and after exiting stage 0 (G_0) are $s_{n-2} \cdots s_0 s_{n-1}$ and $s_{n-2} \cdots s_0 d_{n-1}$, respectively. The former is due to perfect k-shuffle connection and the latter is due to destination tag routing with $t_0 = d_{n-1}$. For stage $i(G_i, 1 \le i \le n-1)$, the channels entering and exiting G_i are $d_{n-1} \ldots d_{n-i} s_{n-i-2} \ldots s_1 s_0 s_{n-i-1}$ and $d_{n-1} \ldots d_{n-i} s_{n-i-2} \ldots s_1 s_0 d_{n-i-1}$, respectively. The former is due to butterfly connection β^k_{n-i}, and the latter is due to destination tag routing with $t_i = d_{n-i-1}$. As the address pattern is changed from $s_{n-1} \ldots s_0$ to $d_{n-1} \ldots d_0$, s_j is always replaced by d_j for all $0 \le j \le n - 1$. Note that for a k-ary m-cube cluster, there are $(n - m)$ fixed variables and m free variables in the definition of the cluster. For communication within each cluster, it implies that $s_i = d_i$ for all those i of fixed variables and the number of free variables m is unchanged for a given k-ary m-cube clusters. Thus, the number of channels between any two adjacent stages of an m-cube cluster remains k^m.

Now we will show that the channels from different clusters (k-ary cubes) are always distinct. Suppose that a channel is shared by two different clusters. The channel address is thus the same from both clusters. This implies that two different clusters have the same fixed variables, a contradiction. $\square$

3. Here we assume that there is no shared flit buffer in each switch; otherwise, the traffic in each cluster may affect other clusters.

In a more general case, when k is a power of 2, the restriction of k-ary cubes can be relaxed to binary cubes. From the proof of Lemma 1, we can immediately obtain the following result.

THEOREM 2. *A cube unidirectional MIN with $N = k^n$ nodes, where $k = 2^j$ for some j, can be partitioned into contention-free and channel-balanced disjoint "binary" cubes.*

Fig. 14 shows the partitioning of an eight-node cube MIN into three contention-free and channel-balanced binary cube clusters: 0XX, 1X0, and 1X1.

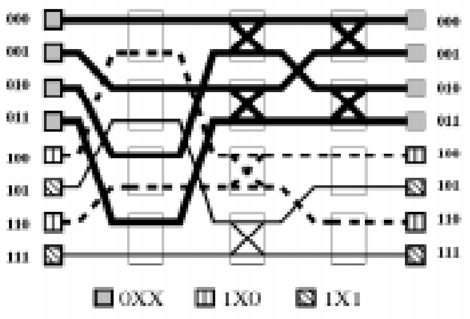


Fig. 14. An eight-node cube MINs is partitioned into three contention-free and channel-balanced binary cube clusters.

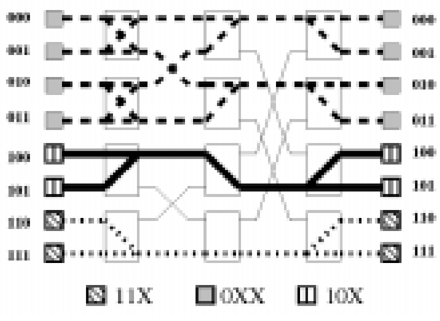
For the case of butterfly MINs, we have the following theorem.

THEOREM 3. *A butterfly unidirectional MIN with $N = k^n$ nodes may not be partitioned into contention-free k-ary cubes.*

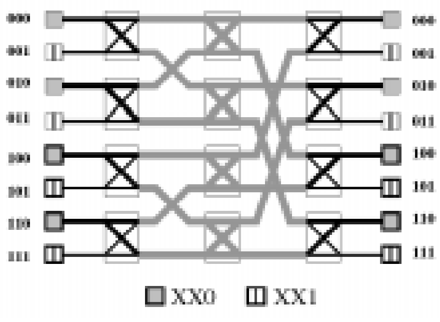
PROOF. Consider a source and destination pair: $s_{n-1} \ldots s_0$ and $d_{n-1} \ldots d_0$. As the address pattern is changed from $s_{n-1} \ldots s_0$ to $d_{n-1} \ldots d_0$, s_j is always replaced by d_{j+1} for $0 \le j \le n-2$ and s_{n-1} is replaced by d_0. Thus, a free variable may be replaced by a fixed variable, implying the number of channels is reduced at that stage. If a fixed variable is replaced by a free variable, this implies that more channels are used by that cluster and the channels may be shared with other clusters. $\square$

Fig. 15 demonstrates the partitioning of an eight-node butterfly MIN into different binary cube clusters. In Fig. 15a, there are three contention-free clusters: 0XX, 10X, and 11X. In all three clusters, the number of channels is reduced to half in some stages. In Fig. 15b, there are two four-node clusters: XX0 and XX1. Both clusters share the use of eight channels.

Due to the nature of butterfly interconnection in BMINs and the analogy to fat tree, we have the following theorem for the partitionability of BMINs. The proof is quite trivial and is ignored here.



(a) reduced channels



(b) shared channels

Fig. 15. An eight-node butterfly MIN is partitioned into different binary clusters.

THEOREM 4. *A butterfly BMIN with $N = k^n$ nodes can be partitioned into contention-free and channel-balanced disjoint "base" k-ary cubes.*

Note that in BMINs, there are many routing paths between some source and destination pairs. Furthermore, if the maximum value of *FirstDifference* between any two nodes in a cluster is t, the maximum number of channels that can be used for a given cluster is k^t at some adjacent stages. Thus, channels are likely to be shared by different clusters. However, due to the existence of multiple routing paths, the performance degradation due to channel contention is not as serious as traditional unidirectional MINs.

5 SIMULATION EXPERIMENTS

To study the performance of four variations of switch-based MINs under different network workloads, we simulate 64-node multistage networks consisting of 4×4 switches. Thus, each network consists of three stages with 16 switches per stage. For all four types of MINs, each node is connected to the network by a pair of unidirectional channels. All channels have the same bandwidth: 20 flits/μsec. Each input channel in a switch has a buffer the size of a single flit. The switches operate asynchronously, but synchronize to simultaneously transmit all of the flits in a *worm*, the portion of a packet in the network. Each node generates packets at time intervals chosen from a negative exponential distribution. Each message has an equal probability of being one packet between eight to 1,024 flits. Messages generated are queued at the source node and enter the network according to the FCFS policy. Messages arriving at a destination node are immediately consumed.

The performance of the network under a given workload and routing algorithm is measured in terms of two main quantities: average communication latency and average sustainable network throughput. The *latency of a message* is the time that elapses between the source node making the message available for transmission and the destination node having the message available for use. The *throughput of a network* is the number of flits delivered per unit of time. It is measured as a percent of the maximum theoretical throughput, which would be achieved if every channel continuously transmitted flits. The throughput is considered sustainable when the number of messages queued at their source nodes does not exceed some small limit, 100 in the simulations.

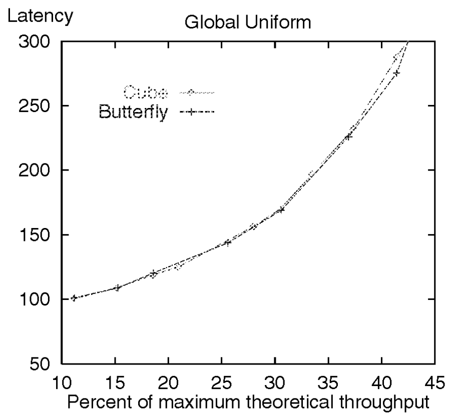
5.1 Network Traffic Patterns

The pattern of message traffic, i.e., the destinations to which messages are sent, has a great impact on the performance of the network. Four message traffic patterns are considered in the simulation: *uniform*, *x% nonuniform*, *perfect k-shuffle*, and *i*th *butterfly*. In our simulation experiments, we consider two different clustering of nodes: global (one cluster with 64 nodes) and cluster-16 (four binary four-cube clusters). For the case of cube networks, four clusters are 0XX, 1XX, 2XX, and 3XX to guarantee channel-balanced partitioning. For the case of butterfly networks, a *channel-reduced* clustering has four clusters: 0XX, 1XX, 2XX, and 3XX; and a *channel-shared* clustering has four clusters: XX0, XX1, XX2, and XX3. Note that in the channel-reduced clustering, the number of channels is reduced from 16 to four, and in the channel-shared clustering, the number of channels is increased from 16 to 64.

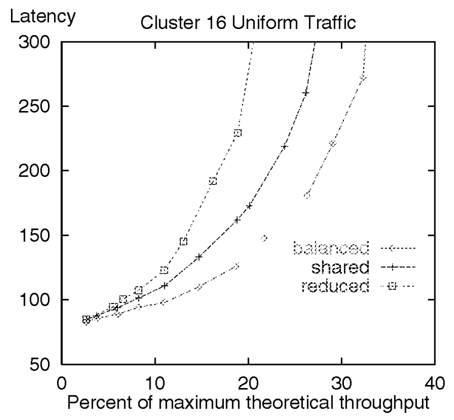
In the uniform traffic pattern, a message generated by each node is sent to any of the other nodes within the same cluster with the same probability. For nonuniform traffic, we consider the first node in each cluster as a hot node, which receive x% more packets than other nodes. Precisely, let $y = Nx$. The probability that the hot node will be selected as a destination is $(1 + y)/(N + y)$ and the probability for each other node is $1/(N + y)$, where N is the total number of nodes in the cluster. The purpose of having such a nonuniform traffic pattern is to observe the effects due to hot spots [20]. Two permutation patterns, perfect k-shuffle and butterfly, are used to observe the network contention due to these special traffic patterns.

5.2 Cube MIN vs. Butterfly MIN

First, we compare the performance between cube unidirectional MINs and butterfly unidirectional MINs with the same workload. Fig. 16 shows the performance between the 64-node cube TMIN and the 64-node butterfly TMIN under global uniform and cluster-16 uniform workloads. For the global uniform traffic, there is no difference between their performance as expected because the whole system is one partition. For the cluster-16 uniform traffic, the communication interference between four clusters in the butterfly TMIN degrades the system performance as shown in Fig. 16b. As expected, the channel-reduced clustering in the butterfly TMIN provides the worst performance. Simulation experiments for the cluster-32 uniform traffic and for DMINs and for VMINs were also conducted. The cube interconnection also showed performance improvement over the butterfly interconnection.

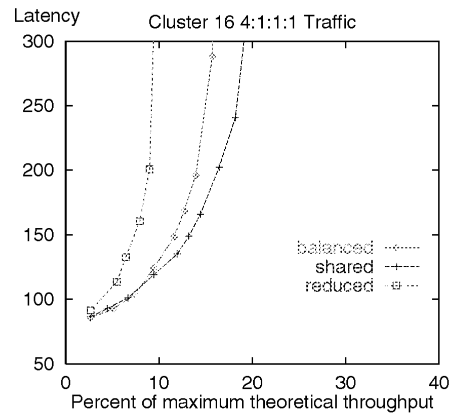


(a)

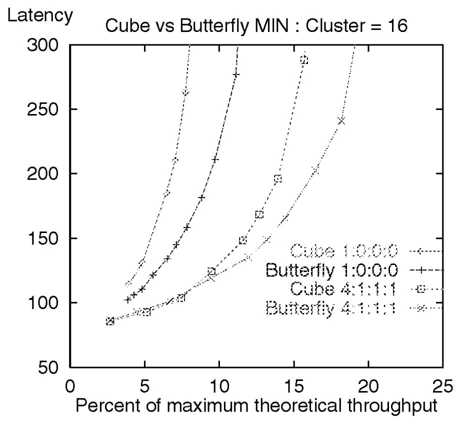


(b)

Fig. 16. The performance of cube and butterfly TMINs under (a) global uniform and (b) cluster-16 uniform workloads.



(a)



(b)

Fig. 17. The performance of cube and butterfly TMINs with four 16-node clusters: (a) ratio 4:1:1:1 and (b) ratios 1:0:0:0 and 4:1:1:1.

We then try to understand the impact of channel-shared partitioning on butterfly networks. Intuitively, the channel-shared partitioning will be beneficial if some clusters generate more traffic than other clusters. Let a:b:c:d be the relative ratio of average network traffic generated from four 16-node clusters. Within each cluster, the network traffic still follows uniform distribution. Fig. 17a shows the results of 4:1:1:1 ratio among four 16-node clusters. In this case, the channel-shared partitioning of the butterfly TMIN provides the best performance. The channel-reduced partitioning of the butterfly TMIN has the worst performance, as expected. Fig. 17b further compares the channel-balanced partitioning of the cube TMIN with the channel-shared partitioning of the butterfly TMIN for two different ratios: 1:0:0:0 and 4:1:1:1. The channel-shared partitioning of the butterfly TMIN always has a better performance. The ratio 1:0:0:0 provides a smaller maximum network throughput because only one cluster of 16 nodes is able to generate network traffic.

The channel-shared partitioning of the butterfly TMIN will provide a better performance than the cube TMIN when the network traffic within each cluster is quite different. However, the channel-shared partitioning in butterfly MINs limits the partitionability of nodes than that of cube MINs. Furthermore, the channel interference among different clusters make the performance of applications in each cluster more difficult to predict. Thus, in the following discussions, we only consider cube interconnection for TMINs, DMINs, and VMINs.

5.3 Comparison of Different Networks

We then compare the performance of three unidirectional cube MINs: TMINs, DMINs, and VMINs, and the butterfly BMINs under different network workloads.

5.3.1 Uniform Traffic Pattern

Fig. 18 shows the performance of four different networks with respect to two uniform workloads: global uniform and cluster-16 uniform.

The TMIN performs the worst in both cases because it has only one path between each pair of source and destination. The DMIN performs consistently the best because the DMIN always has two choices at each outgoing port. The physical channel bandwidth available between stages in the DMIN always doubles than that of the TMIN and VMIN. Surprisingly, the performance of the VMIN is always slightly better than that of the BMIN. This can be explained that the channel blocking probability in the VMIN is reduced, while in the BMIN, there is only one path from the turnaround point to

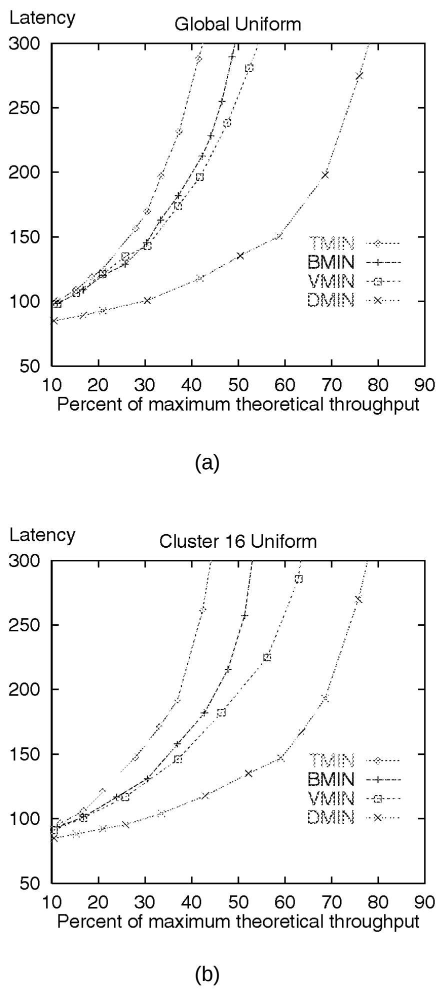


Fig. 18. The performance of four networks under (a) global uniform and (b) cluster-16 uniform workloads.

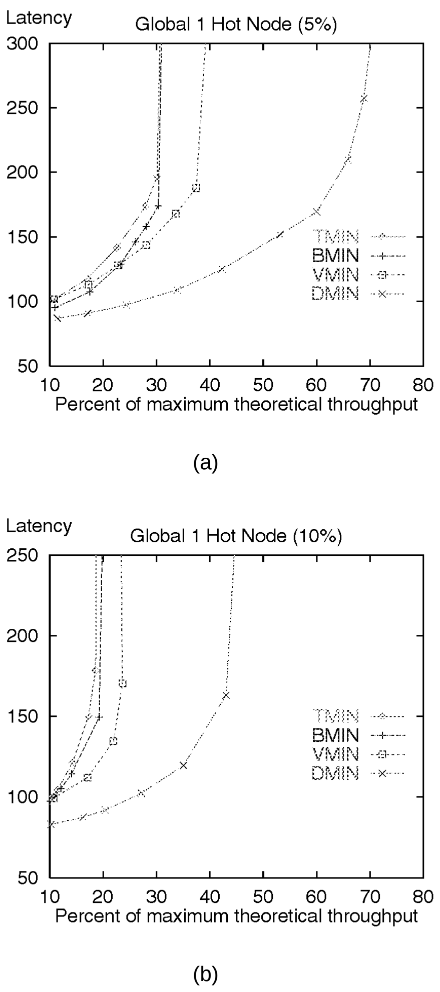


Fig. 19. The performance of four networks under the global hot spot workload: (a) 5% more traffic (b) 10% more traffic.

the destination. A similar relative performance difference was also observed for the cluster-32 uniform workload.

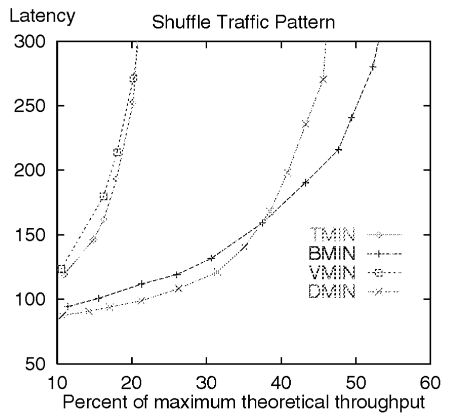
5.3.2 Hot Spot Traffic Pattern

Fig. 19 shows the performance when there are hot spots in the network for the case of a global cluster. As shown in the figure, all four networks are congested as indicated by their reduced network throughput comparing with Fig. 18a. The DMIN always has the best performance due to the same reason as in the uniform traffic case. The performance degradation for the DMIN is quite small (from 78% to 70%) as demonstrated in Fig. 18a and Fig. 19a when the hot node traffic is 5% more. When the hot node traffic is increased to 10% more, the maximum network throughput is reduced to 45% (DMIN in Fig. 19b). Since the TMIN does not provide any alternate routing path, it has the worst performance. Note that the performance difference between the TMIN and BMIN is quite small because the path down the tree is unique in the BMIN, while both the DMIN and VMIN have alternate paths all the way. The performance of the VMIN is expected to be better if there are additional virtual channels. The above relative performance difference among four different networks is the same for cluster-16 and cluster-32 partitionings.

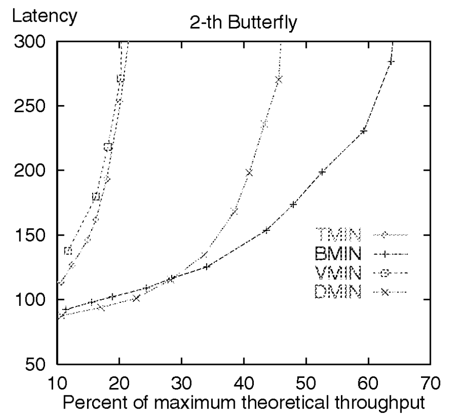
5.3.3 Permutation Traffic Patterns

Fig. 20 shows the performance under two special permutation traffic patterns. For both permutation patterns, channel contention may occur in all four networks. Both the TMIN and the VMIN have a poor performance, because some channels have to be shared by four source and destinations pairs. The VMIN has worse performance than that of the TMIN because the flit-level sharing of channels is based on round-robin scheduling, where the objective is to maintain fairness. Thus, all contending packets have similar long delays. Note that the physical channel bandwidth is the same for both the TMIN and VMIN.

Both the DMIN and the BMIN demonstrate a better performance due to the existence of multiple routing paths. When the workload is heavy, the BMIN provides the best performance. In the DMIN (dilation two), the channel contention is still possible due to the competition of four packets over some channels. In the BMIN, theoretically, all source and destination pairs can be transmitted simultaneously without contention if the forward channel is properly chosen. Even if not, the probability of contention is quite small due to the existence of multiple routing paths.



(a)



(b)

Fig. 20. The performance of four networks under (a) shuffle permutation and (b) 2-th butterfly permutation traffic patterns.

6 Conclusions and Future Work

This paper has described four switch-based wormhole networks. Some of these networks, such as TMINs and BMINs, have been adopted in many commercial parallel computers. This paper provides a comprehensive performance comparison among them. Among these four MINs, both DMINs (dilation two) and BMINs have a similar hardware and packaging complexity. In terms of individual switch design, VMINs, DMINs, and BMINs have a similar hardware complexity.[4] From our simulation results, we conclude that a DMIN (dilation two) is the most cost effective design comparing with the corresponding popular BMIN (or fat tree) for most of the network traffic patterns. Note that our conclusion is based on the following conditions.

- Each node has one input port and one output port—the so-called one-port communication architecture.
- The network cycle time is the same for all network architectures.
- There is only one flit buffer for each channel.

The BMIN does have some advantages. It has flexible density of interconnect, supports communication locality well, does not require wraparound connections, and can provide a good performance if the application specific routing paths can be carefully scheduled. We have shown that although those Delta-class MINs are topologically and functionally equivalent, they are not equivalent in terms of network partitionability. Our additional work has shown that the Omega network and the cube network have the same network partitionability; while the baseline network and the butterfly network have a similar network partitionability.

There are many directions for future research. Some may argue that our performance comparison among four different networks is unfair. However, it is difficult to make a truly fair comparison. We can double the channel bandwidth for both the TMIN and the VMIN, which becomes the same as the DMIN and the BMIN. Clearly, the performance of the TMIN and the VMIN will be significantly improved. However, this may not be fair to the other two networks because doubling the channel bandwidth implies that the input and output bandwidth to each node is also doubled. More simulation experiments need to be conducted to study the impact due to long, short, and bimodal message sizes, hot spot contention in different clusters, other nonuniform traffic patterns, other network and switch sizes, and other partitioning of clusters. Some other variations of multiple-path MINs, such as multibutterflies [31], extra-stage MINs, BMINs with virtual channels, and VMINs with more than two virtual channels, deserve further comparison. More detailed cost and hardware design study of these networks is another interesting area for further research. We have studied software support of multicast communication in BMINs [32]. Both hardware and software multicast support for these networks is another interesting research area.

4. The hardware design complexity of switches in BMINs may be higher due to more choices of outgoing ports for a given input packet.

Acknowledgments

We thank the anonymous referees for their constructive comments and suggestions. This work was supported in part by U.S. National Science Foundation Grants No. MIP-9204066 and MIP-9528903, and U.S. Department of Energy Grant No. DE-FG02-93ER25167.

References

[1] L.M. Ni and P.K. McKinley, “A Survey of Wormhole Routing Techniques in Direct Networks,” *Computer*, vol. 26, no. 2, pp. 62-76, Feb. 1993.
[2] BBN Advanced Computers Inc., *Inside the GP1000*, Cambridge, Mass., 1989.
[3] BBN Advanced Computers Inc., *Inside the TC2000 Computer*, Cambridge, Mass., 1990.
[4] D.M. Das and J.R. Jump, “Analysis and Simulation of Buffered Delta Networks,” *IEEE Trans. Computers*, vol. 30, no. 4, pp. 273-282, Apr. 1981.
[5] C. Kruskal and M. Snir, “The Performance of Multistage Interconnection Networks for Multiprocessors,” *IEEE Trans. Computers*, vol. 32, no. 12, pp. 1,091-1,098, Dec. 1983.
[6] J. Ding and L.N. Bhuyan, “Finite Buffer Analysis of Multistage Interconnection Networks,” *IEEE Trans. Computers*, vol. 43, no. 2, pp. 243-247, Feb. 1994.
[7] W.J. Dally and C.L. Seitz, “The Torus Routing Chip,” *J. Distributed Computing*, vol. 1, no. 3, pp. 187-196, 1986.
[8] W.J. Dally, “Virtual Channel Flow Control,” *Proc. 17th Int’l Symp. Computer Architecture*, pp. 60-68, May 1990.

[9] L.M. Ni, "Issues in Designing Truly Scalable Interconnection Networks," *Proc. 1996 ICPP Workshop Challenges for Parallel Processing*, pp. 74-83, Aug. 1996.

[10] L.R. Goke and G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessing Systems," *Proc. First Int'l Symp. Computer Architecture*, pp. 21-28, 1973.

[11] J.H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. Computers*, vol. 30, no. 10, pp. 771-780, Oct. 1981.

[12] C.L. Wu and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers*, vol. 29, no. 8, pp. 694-702, Aug. 1980.

[13] H.J. Siegel, W.G. Nation, C.P. Kruskal, and L.M. Napolitano Jr., "Using the Multistage Cube Network Topology in Parallel Supercomputers," *Proc. IEEE*, vol. 77, pp. 1,932-1,953, Dec. 1989.

[14] D.J. Kuck, E.S. Davidson, D.H. Lawrie, and A.H. Sameh, "Parallel Supercomputing Today and the Cedar Approach," *Science*, vol. 231, pp. 967-974, Feb. 1986.

[15] H.J. Siegel, L.J. Siegel, F.C. Kemmerer, P.T. Mueller, Jr., H.E. Samlley, Jr., and S.D. Smith, "PASM: A Partitionable SIMD / MIMD System for Image Processing and Pattern Recognition," *IEEE Trans. Computers*, vol. 30, no. 12, pp. 934-947, Dec. 1981.

[16] G.F. Pfister et al., "An Introduction to the IBM Research Parallel Processor Prototype (RP3)," *Experimental Parallel Computing Architectures*, J.J. Dongarra, ed., pp. 123-140, Amsterdam: Elsevier Science Publishers B.V., 1987.

[17] A. Gottlieb, "An Overview of the NYU Ultracomputer Project," *Experimental Parallel Computing Architectures*, J. Dongarra, ed., pp. 25-95, North Holland, 1987.

[18] R.D. Rettberg, W.R. Crowther, P.P. Carvey, and R.S. Tomlinson, "The Monarch Parallel Processor Hardware Design," *Computer*, vol. 23, no. 4, pp. 18-30, Apr. 1990.

[19] N. Koike, "NEC Cenju-3: A Microprocessor-Based Parallel Computer," *Proc. Eighth Int'l Symp. Parallel Processing*, pp. 396-401, Apr. 1994.

[20] G. Pfister and A. Norton, "Hot Spot Contention and Combining in Multistage Interconnect Networks," *IEEE Trans. Computers*, vol. 34, no. 10, pp. 943-948, Oct. 1985.

[21] M.E. Becker and J. Thomas F. Knight, "Fast Arbitration in Dilated Routers," *Proc. First Int'l Workshop Parallel Computer Routing and Comm.*, K. Bolding and L. Snyder, eds., pp. 16-30, Springer-Verlag, May 1994.

[22] A.A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers," *Proc. Hot Interconnects '93*, Aug. 1993.

[23] Cray Research, Inc., *CRAY T3D System Architecture Overview*, Chippewa Falls, Wis., 1993.

[24] W.J. Dally, L.R. Dennison, D. Harris, K. Kan, and T. Xanthopoulus, "The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers," *Proc. First Int'l Workshop Parallel Computer Routing and Comm.*, K. Bolding and L. Snyder, eds., pp. 241-255, Springer-Verlag, May 1994.

[25] J. Bruck, R. Cypher, L. Gravano, A. Ho, C.T. Ho, S. Konstantinidou, E. Upfal, S. Kipins and M. Snir, "Survey of Routing Issues for the Vulcan Parallel Computer," Technical Report RJ8839, IBM Research Division, June 1992.

[26] I.D. Scherson and C.K. Chien, "Least Common Ancestor Networks," *Proc. Seventh Int'l Symp. Parallel Processing*, pp. 507-513, Apr. 1993.

[27] C.E. Leiserson et al., "The Network Architecture of the Connection Machine CM-5," *Proc. ACM Symp. Parallel Algorithms and Architectures*, pp. 272-285, San Diego, 1992.

[28] Meiko Limited, *Computing Surface: CS-2 Communications Networks*, Waltham, Mass., 1993.

[29] C.B. Stunkel et al., "Architecture and Implementation of Vulcan," *Proc. Eighth Int'l Symp. Parallel Processing*, pp. 268–274, Apr. 1994.

[30] C.B. Stunkel, D.G. Shea, D.G. Grice, P.H. Hochschild, and M. Tsao, "The SP1 High-Performance Switch," *Proc. 1994 Scalable High Performance Computing Conf.*, pp. 150-157, May 1994.

[31] C.E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. Computers*, vol. 34, no. 10, pp. 892-901, Oct. 1985.

[32] F.T. Leighton and B.M. Maggs, "Fast Algorithms for Routing around Faults in Multibutterflies and Randomly-Wired Splitter Networks," *IEEE Trans. Computers*, vol. 41, no. 5, pp. 578-587, May 1992.

[33] H. Xu, Y. Gui, and L.M. Ni, "Optimal Software Multicast in Wormhole-Routed Multistage Networks," *Proc. Supercomputing '94*, pp. 703-712, Nov. 1994.

Lionel M. Ni (S'78-M'81-SM'87-F'94) received his Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1980. In 1981 he joined the faculty of the Department of Computer Science, Michigan State University, East Lansing, where he is currently a professor. His current research interests include computer architecture, parallel processing, high-speed networks, web-based software tools, voice over the Internet, and distributed computing.

Dr. Ni is serving as a member of the editorial boards of IEEE Transactions on Parallel and Distributed Systems and Journal of Information and Science and Engineering. He served as an IEEE Computer Society Distinguished Visitor from 1985 to 1988, an editor for IEEE Transactions on Computers from 1992 to 1996, a subject area editor for the Journal of Parallel and Distributed Computing from 1987 to 1994, a program evaluator for the Computer Sciences Accreditation Commission from 1989 to 1993, and the program director of the U.S. National Science Foundation Microelectronic Systems Architecture Program from 1995 to 1996.

He is a member of the Association for Computing Machinery and SIAM. He was elevated to the rank of fellow of IEEE in 1994 for his contributions to parallel processing and distributed systems. He served on program committees of many conferences; was program chair of the Fifteenth IEEE Annual International Computer Software and Applications Conference, the 1994 International Conference on Parallel and Distributed Systems; and the 1997 IEEE High Performance Computer Architecture Symposium; and is program chair of the 1997 International Conference on Computer Communications and Networks. He has received a number of awards for authoring outstanding papers and also won the Michigan State University Distinguished Faculty Award in 1994.

Yadong Gui received the B.S. degree in 1965, and the M.S. and Ph.D. degrees in computer science from the Xian Science and Technology University, P.R. China, in 1980 and 1986, respectively. During 1993 to 1994, he was a visiting scholar at Michigan State University, East Lansing, MI, USA.

Dr. Gui is a professor of the Institute of Computing Technology in the Chinese Academy of Sciences. His research interests include multiprocessor architectures, interconnection networks, and networks of workstations.

Sherry Q. Moore is a PhD candidate in the Computer Science Department at Michigan State University, E. Lansing, MI. She is currently a member of the technical staff at Sun Microsystems. Prior to that she worked as a system architect at Intel Corporation. Her current research focuses on distributed computing on heterogeneous platforms. She received the B.S. degree in computer engineering from Beijing University of Posts and Telecommunication, P.R. China, in 1991, and the M.S. degree in computer science from Michigan State University in 1994.