A Framework for Designing Deadlock-Free Wormhole Routing Algorithms

Rajendra V. Boppana and Suresh Chalasani

Abstract—This paper presents a framework to design fully-adaptive, deadlock-free wormhole algorithms for a variety of network topologies. The main theoretical contributions are (a) design of new wormhole algorithms using store-and-forward algorithms, (b) a sufficient condition for deadlock free routing by the wormhole algorithms so designed, and (c) a sufficient condition for deadlock free routing by these wormhole algorithms with centralized flit buffers shared among multiple channels. To illustrate the theory, several wormhole algorithms based on store-and-forward hop schemes are designed. The hop-based wormhole algorithms can be applied to a variety of networks including torus, mesh, de Brujin, and a class of Cayley networks, with the best known bounds on virtual channels for minimal routing on the last two classes of networks. An analysis of the resource requirements and performances of a proposed algorithm, called negative-hop algorithm, with some of the previously proposed algorithms for torus and mesh networks is presented.

Index Terms—Adaptive routing, Cayley networks, de Bruijn networks, deadlocks, design techniques, multicomputer networks, mesh networks, performance evaluation, wormhole routing.

1 Introduction

Many recent experimental and commercial parallel computers [1], [3], [7], [25], [30], [32], [36], [41] use direct networks for low latency, high bandwidth interprocessor communication. A typical direct network is the k-ary n-cube network, which has an n-dimensional grid structure with k nodes (processors) in each dimension such that every node is connected to two other nodes in each dimension by direct communication links.

The performance of a multicomputer network depends on the *switching technique* and the *routing algorithm* used. Possible switching techniques are the virtual cut-through [27], store-and-forward [22], and wormhole [13]. The *wormhole* (WH) switching technique has been widely used in the recent multicomputers [32], [30], [36]. In the WH technique, a packet is divided into a sequence of fixed-size units of data, called *flits*. If a communication channel transmits the first flit of a message, it must transmit all the remaining flits of the same message before transmitting flits of another message. The main advantages of wormhole switching are low memory requirements in routers and pipelined data movement in the absence of contention. The main disadvantage of wormhole switching is channel congestion, since a blocked message does not relinquish the communication channels it has already acquired. The virtual cut-through, VCT, and store-and-forward, SAF, switching techniques require more storage in nodes but have less channel contention.

- *R.V. Boppana is with the Division of Computer Science, The Univesity of Texas, San Antonio, San Antonio, TX 78249-0664. E-mail: boppana@ringer.cs.utsa.edu.*
- *S. Chalasani is with the Department of Electrical and Computer Engineering, Univesity of Wisconsin-Madison, Madison, WI 53706-1691. E-mail: suresh@cauchy.ece.wisc.edu.*

Manuscript received May 2, 1994; revised June 27, 1995.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number D95077.

Some of the most important issues in the design of a routing algorithm are high throughput, low-latency message delivery, avoidance of deadlocks, livelocks, and starvation [17]. In this study we consider only *minimal* routing algorithms as per which a message always moves closer to its destination with each hop taken. Livelocks can be avoided with minimal routing, and starvation can be avoided by allocating resources such as communication channels and buffers in FIFO order. Ensuring deadlock-freedom depends on the design of the routing algorithm.

A routing algorithm that provides messages with multiple paths to use to reach their destinations is an *adaptive* routing algorithm. Minimal fully-adaptive algorithms do not impose any restrictions on the choice of shortest paths to be used in routing messages; in contrast, partially-adaptive minimal algorithms allow only a subset of available minimal paths in routing messages. The well-known e-cube routing algorithm [13], [43] is an example of *non*-adaptive routing algorithms, since it has no flexibility in routing messages.

Many researchers are investigating suitable adaptive wormhole and virtual cut-through algorithms for high-performance and fault-tolerant routing in k-ary n-cube based tori and meshes [4], [5], [8], [9], [12], [14], [19], [28], [31], [34], [35], [38], [40], and other networks [18], [33]. Most of the recent results are on the design of adaptive wormhole algorithms using as few virtual channels as possible [4], [9], [14], [16], [19]. Incorporating adaptivity may not always improve the throughput and average message latency [6], [19]. Further, multiple virtual channels could be multiplexed on a single physical channel using additional flit buffers and multiplexers to improve performance [11], [21], [25].

The work on designing wormhole routing algorithms is done largely independent of the results developed for store-

and-forward switched computer networks [20], [22]. There are no general results which show the applicability of SAF algorithms to derive corresponding WH algorithms without compromising adaptivity and deadlock-freedom. Further, with the exception of a few results [13], [18], [33], the current results on wormhole algorithms are targeted to k-ary n-cube torus and mesh networks.

Based on these observations, it is appropriate to ask the following questions. Can we apply the routing algorithms for SAF computer networks to WH multicomputer networks? Furthermore, how can we develop WH routing algorithms that can be applied to a variety of networks including the k-ary n-cube based meshes and tori, de Brujin [39] and n-star [2]? What are the performance implications of the routing algorithms so derived?

To address these issues, we present a general result to show that a class of store-and-forward routing algorithms can also be used, with appropriate modifications, for WH routing. We believe that this result unlocks the potential of a large number of results developed for computer networks in the past two decades. We provide sufficient conditions for deadlock free routing by these wormhole algorithms. We also provide a sufficient condition for sharing flit buffers among multiple channels without creating deadlocks.

As an example of our technique, we derive several deadlock-free, fully-adaptive WH routing algorithms from SAF algorithms. These algorithms are based on the number of hops taken by messages, and are called *hop* schemes. For k-ary n-cube networks, hop schemes require more virtual channels than some of the recently proposed wormhole algorithms [4], [14], [40]. But hop schemes provide deadlock free routing even when flit buffers are shared among multiple channels. We show in our performance comparisons with other algorithms that this ability of hop schemes makes them competitive for many practical network sizes. Furthermore, the hop schemes are versatile, and can be used for WH switched networks with any topology. To illustrate this, we provide minimal, deadlock-free, and fully-adaptive algorithms with the best known bounds on virtual channel requirements for the de Brujin and n-star networks.

The rest of the paper is organized as follows. Section 2 presents the result on developing WH routing algorithms from SAF algorithms. Section 3 presents WH hop schemes and their variants. Section 4 applies the results to develop fully-adaptive WH routing algorithms for de Brujin and n-star networks. Section 5 compares the proposed schemes with the adaptive WH routing algorithms proposed in the literature. Section 6 concludes the paper with directions for future research.

2 Application of SAF Algorithms for WH Routing

In this section, we describe a method to design new wormhole routing algorithms from store-and-forward algorithms. We also present a sufficient condition under which the new wormhole algorithms are deadlock free.

First, we introduce some terminology. Each node of the interconnection network is a processor-memory-router element and is given a distinct address. We assume that the links of the network are bidirectional, which can be implemented using two unidirectional (simplex) physical communication channels in opposite directions. The physical channels, buffers, virtual channels, and messages originating from a node can be given unique numbers based on the address of the node. Unless otherwise indicated, the number of virtual channels are specified per physical channel.

Let N denote the set of nodes in the network and pc denote the set of all physical channels in the network. In a SAF network, $\boldsymbol{b}_i$ denotes the set of class i buffers, and $\boldsymbol{b} = \cup_{i=1}^{m} \boldsymbol{b}_i$ is the set of all buffers in the network, where m is the number of buffer classes used. Let Class(b) and Channel(b, b') denote, respectively, the class of buffer b and the physical channel connecting the nodes to which b and b' belong. In a WH network, c_i denotes the set of class i virtual channels, and $\boldsymbol{c} = \cup_{i=1}^{m} c_i$ is the set of all virtual channels in the network, where m is the number of virtual channel classes used. Let *Channel*(c) denote the physical channel on which the virtual channel c is simulated, and *Class*(c) denote the class of c.

2.1 Deadlock Free Routing Concepts

We assume that a message which reached its destination does not require any more network resources—buffers in SAF and communication channels in WH—and is consumed in a finite amount of time. Therefore, the issue of deadlocks is concerned with the messages that have acquired some network resources and need more resources to reach their destinations.

In WH routing, communication channels are the resources for which messages compete. A single physical channel between adjacent nodes may not provide deadlock-free routing in multicomputer networks such as k-ary n-cube based meshes and tori. One solution is to provide a sufficient number of virtual channels and devise a suitable routing algorithm [13]. Multiple virtual channels between a pair of adjacent nodes is provided by multiplexing the bandwidth of the single physical channel available.

A wormhole routing algorithm specifies two relations on virtual channels: routing relation, R, and selection relation, δ. The routing relation determines which paths and channels are suitable, for example, for deadlock-free routing, for the next hop of a message. The selection relation δ uses additional criteria such as channel congestion and chooses one of the channels indicated by R. The issue of deadlocks is addressed in the design of R, leaving specification of δ for performance improvements only [14]. In this paper, we use routing relation and routing algorithm synonymously.

In SAF routing, multiple classes of buffers are used to avoid deadlocks and improve performance. All of the above discussion applies to SAF routing, when virtual channels are replaced by buffers.

Let $\boldsymbol{r}$ denote the set of resources (buffers in SAF and virtual channels in WH) used by the routing relation R. We use the *maximal resource dependency graph* of the routing relation R. The maximal resource dependency graph, henceforth *resource graph*, of R is obtained as follows. The vertices of resource graph are the resources (buffers or virtual channels); there is a directed edge from vertex $r1$ to $r2$ if a message can use $r2$ immediately after using $r1$. For deadlock

proofs, we show that maximal dependency graphs are acyclic. We consider only minimal (shortest-path) routing of messages. Minimal routing avoids livelocks and minimizes the bandwidth used per message. We avoid starvation by assigning resources to waiting messages on a FCFS basis.

2.2 Construction of New Wormhole Algorithms

In this section, we establish the correspondence between SAF and WH routing algorithms. Fig. 1 illustrates construction of a wormhole node from an SAF node. In SAF routing, buffers in nodes are the critical resources. Deadlocks in SAF routing are avoided by partitioning the buffers into several classes and placing constraints on the set of buffer classes a message can occupy in each node. This is known as the *buffer reservation* technique [22].

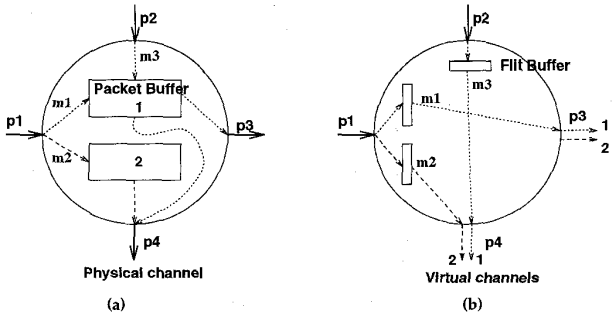


Fig. 1. Example of a wormhole router construction from a store-and-forward router. The SAF note in (a) has two input, $p1$ and $p2$, and two output $p3$ and $p4$, physical channels and two packet butters, $b1$ and $b2$. The paths of three messages, $m1$, $m2$, and $m3$, through the SAF node are shown. In the corresponding wormhole node in (b), two virtual channels are simulated on each input and output physical channel. For clarity, only output virtual channels are shown. The paths of the three messages through the WH node are based on the packet butter and the output physical channels used int he SAF node. The flit butter used to store a flit of a messagae, for example, $m1$, is dependent on the virtual channel used by $m1$ on $p1$.

In the SAF algorithms based on buffer reservations, each message is given a class, and a message of class i occupies a buffer of class i. A message takes hops from one buffer to another until it occupies a buffer in its destination node, at which point it awaits consumption. Then, the routing relation, S, for an SAF algorithm is from $\boldsymbol{b} \times N$ to $\boldsymbol{b}$. Hops allowed are given by the elements of S. The element $(b1, y, b2) \in S$ represents a hop allowed from buffer $b1$ to $b2$ by a message destined to y.

The process of designing a wormhole algorithm, W, from an SAF algorithm, S, consists of two steps: specification of c, the set of virtual channels, and W, the routing relation from $c \times N$ to c.

1) Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ be classes of buffers occupied by messages before reaching their destinations in the SAF algorithm. Then, for the WH algorithm, on each physical channel in the network, we provide virtual channels of classes $c_1, \ldots, c_m$ and the corresponding flit-buffers. Fig. 1 shows this for $m = 2$. Therefore, the set of virtual channels in the entire network is $c \supseteq \{1,\ldots, m\} \times pc$. We also include injection channels and consumption channels of all nodes in C.
2) Let $(b_1, y, b_2) \in S$, a hop from buffer b_1 to b_2 by a message destined to y in the SAF routing. Then, $(c', y, c1)$, $(c1, y, c'') \in W$, where $Class(b1) = Class(c1)$, $Channel(c1) = Channel(b1, b2)$, c' is any virtual channel simulated for any buffer and physical channel combination used by the message to reach $b1$, and c'' is any virtual channel simulated for any buffer and physical channel combination used by the message after reaching $b2$ (see Fig. 2). If $(b1, y, b2)$ is the first hop of the message in the SAF routing, then c' is inj, the injection channel of the node of $b1$. If $(b1, y, b2)$ is the last hop of the message in the SAF routing, then c'' is $cons$, the consumption channel of the node of $b2$.

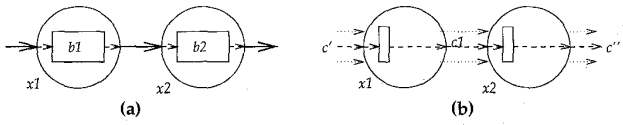


Fig. 2. Illustration of hops in a wormhole algorithm constructed from a store-and-forward algorithm. Part (a) illustrates the hop by a message from packet buffer $b1$ to $b2$ in SAF routing. Part (b) illustrates the corresponding hop by the same message in WH routing. The virtual channel c' provided for the hps by messages from packet buffer $b1$ to node $x2$ is used for the WH hop. The virtual channels c' and c'' are dependent on the hops taken before arriving at node $x1$ and after arriving at node $x2$, respectively. The flit buffer used in node $x1$ is the dedicated flit buffer for c' and the flit buffer used in node $x2$ is the dedicated flit buffer for $c1$. The dotted lines indicate additional virtual channels (flit butters not shown) simulated on each physical channel.

Informally, if the SAF algorithm specifies that a message should occupy a buffer of class $\boldsymbol{b}_i$ at node x and use a channel from a set of physical channels, $\boldsymbol{l}$, to complete the next hop, the corresponding WH algorithm specifies that the message at x should take the next hop using a virtual channel of class c_i on any of the physical channels in $\boldsymbol{l}$.

Suppose a message M is routed from x to y using buffers $b_1, \ldots, b_t$, for hops 1, ..., $t - 1$; b_1 is the buffer occupied at injection and b_t is the buffer occupied at consumption. Therefore, (b_1, y, b_2), (b_2, y, b_3), ..., $(b_{t-1}, y, b_t) \in S$. Then, (inj_x, y, c_1), ..., (c_{t-2}, y, c_{t-1}), $(c_{t-1}, y, cons_y) \in W$, such that $Class(c_i) = Class(b_i)$, $0 < i < t$, and $Channel(c_i) = Channel(b_i, b_{i+1})$; inj_x is the injection channel in node x and $cons_y$ is the consumption channel in node y.

2.3 Sufficient Conditions for Deadlock Free Wormhole Algorithms

The procedure above designs a wormhole routing algorithm, W, from a store-and-forward algorithm, S, with the same degree of adaptivity. However, W need not be deadlock-free. We will provide sufficient conditions for S to yield deadlock-free W.

It is obvious that any routing algorithm for multicomputers should ensure the following:

1) each and every message injected into the network is delivered to its destination, and
2) each message delivered to its destination is removed from the network in finite time.

In addition, the SAF algorithms considered in this paper have the following property.

PROPERTY 1. *The buffer occupied by a message in a given node is dependent only on the buffer occupied in the previous node and the channel used for the hop between the previous node and the present node.*

Consider a message M destined to y and currently occupying buffer b_1 in node x_1. If it moves to buffer b_2 in x_1's neighbor x_2, then each and every message that occupies b_1 (in x_1) and moves to x_2 in the next hop can use b_2. Furthermore, any message that is destined to y and occupying b_1 can move to or wait for b_2 without any restrictions. The routing relations of such SAF algorithms are said to be static.

All dependency graphs used for the rest of the paper actually refer to maximal dependency graphs, which are explained in Section 2.1. We classify the cycles of a dependency graph into two categories: direct and indirect cycles. A direct cycle passes though exactly two vertices. An indirect cycle is an elementary cycle—a cycle such that no vertex is encountered more than once—passing through three or more vertices. Resource graphs do not have self-loops, which are cycles involving only one node.

LEMMA 1. *The maximal dependency graph of a routing algorithm has*

1) *direct cycles if and only if the algorithm has direct deadlocks and*
2) *indirect cycles if and only if the algorithm has indirect deadlocks.*

The lemma presented above is a restatement of the well-known result in operating systems and in computer communications and applies to routing algorithms with static Rs. Routing algorithms with dynamic Rs have deadlocks if and only if instantaneous dependency graphs—formed by taking currently existing dependencies—have cycles. This fact is used in designing many adaptive algorithms [14], [38], [40].

In store-and-forward algorithms, it is feasible to use centralized buffers, which could lead to a direct deadlock—two messages in adjacent nodes block each other's path. The following lemma shows that SAF algorithms with direct deadlocks can be used to construct WH algorithms if certain conditions are met. The scope of the lemma includes SAF algorithms with nonminimal routing.

LEMMA 2. *$\mathbf{W}$ is free of direct and indirect deadlocks if S is free of indirect deadlocks and satisfies any one of the following conditions:*

1) *a message always acquires buffers not used by it before,*
2) *a message does not revisit a node immediately after leaving it, or*
3) *a message never visits the same node twice.*

PROOF. Assume that the channel graph of $\mathbf{W}$ has a cycle: $c_1, \ldots, c_t, c_1$, $t > 1$. Then the buffer graph of S has the following cycle: $b_1, b_2, \ldots, b_t, b_1$, such that the hop on c_i corresponds to the hop $(b_i, y_i, b_{i,\text{mod } t+1})$. Since the given SAF algorithm has no indirect deadlocks, there cannot be indirect cycles in its buffer graph. Hence, $t = 2$. So, indirect cycles do not occur in the channel graph. Now, we show that direct cycles cannot occur in the channel graph if the hypothesis is satisfied.

PART A. Assume that a message never reuses a buffer in the SAF routing. Consider the cycle $c1, c2, c1$ in the channel graph: message $m1$ obtained $c1$ and waits for $c2$ and message $m2$ obtained $c2$ and waits for $c1$. Therefore, the wormhole algorithm allows $m1$ to revisit its current node, via $c2$, immediately after leaving it. In the corresponding SAF routing, $m1$ waits for buffer of $m2$ and vice versa. Furthermore, $m1$ can revisit its current node using the buffer and physical channel used by $m2$. Therefore, by Property 1, $m1$ can use its current buffer on its revisit. This is a contradiction.

PART B. Suppose a message never revisits a node immediately after leaving it. Then it may reuse a buffer after taking two or more hops. But this implies an indirect cycle in the buffer graph, which cannot occur. Therefore, there cannot be cycles in the channel graph.

PART C. This part is a direct consequence of PART A, since a message that never revisits a node does not reuse a buffer. □

COROLLARY 1. *If S ensures that messages acquire buffers in the greater than order, $\succ$, of some partial order on $\mathbf{b}$, then $\mathbf{W}$ is deadlock free.*

PROOF. Since S allocates buffers to messages as per an antisymmetric relation, no message reuses a buffer, and S is free of deadlocks. Therefore, S satisfies the hypothesis of Lemma 2. □

In the next two sections, we consider a few well-known SAF schemes based on the number of hops taken [20] and derive several deadlock-free wormhole algorithms for meshes, tori, de Brujin, and a class of Cayley (star) networks.

3 WORMHOLE HOP SCHEMES

In hop schemes, the class of a message at any time is a function of the hops it has taken up to that point. Depending on the function used, various hop schemes can be designed. In this section, we describe the *negative-hop* (NHOP) scheme, which is based on the NHOP SAF algorithm by Gopal [20], and several variations of the NHOP scheme.

We use the following notation for mesh and torus networks. A (k, n)-torus (also called k-ary n-cube) has n dimensions, $\text{DIM}_0, \ldots, \text{DIM}_{n-1}$, and $N = k^n$ nodes. Each node is uniquely indexed by an n-tuple in radix k. Each node is connected via communication links to two other nodes in each dimension. The neighbors of the node $x = (x_{n-1}, \ldots, x_0)$ in DIM_i are $(x_{n-1}, \ldots x_{i+1}, x_i \pm 1, x_{i-1}, \ldots, x_0)$, where addition and subtraction are modulo k. A link is said to be a wraparound link if it connects two neighbors whose addresses differ by $k-1$ in DIM_i, $0 \le i < n$. A (k, n)-mesh is a (k, n)-torus with the wraparound connections missing. The well-known binary hypercube is the $(2, n)$-mesh. In this paper, we consider (k, n)-torus and (k, n)-mesh networks with small n, large k, and bidirectional links.

3.1 The Negative-Hop Algorithm

The SAF Version. In the negative-hop SAF algorithm [20], the network is partitioned into several subsets, such that no

subset contains two adjacent nodes (this is the graph coloring problem). If C is the number of subsets, then the subsets are labeled 0, 1, ..., $C-1$, and nodes in subset i are labeled (colored) i. A hop is a negative hop if it is from a node with a higher label to a node with a lower label; otherwise, it is a nonnegative hop. A message occupies a buffer of class b_i at an intermediate node if and only if the message has taken exactly i negative hops to reach that intermediate node. If H is the maximum hops taken by a message and C is the number of colors, then the maximum number of negative hops that can be taken by a message is

$$H_N = \lceil H(C-1)/C \rceil. \quad (1)$$

Gopal [20] proves that this SAF routing is deadlock free when $H_N + 1$ classes of buffers are used.

The WH Version. The number of virtual channels used in the negative-hop (NHOP) wormhole algorithm is proportional to the maximum number of negative hops a message can take. If m is the maximum negative hops taken by a message, then up to $m+1$ virtual channels, one for each of virtual channel classes $c_0, \ldots, c_m$, are simulated on each physical channel. Every message uses a virtual channel of class c_0 for its first hop. Further, the class of a message increases by one after each negative hop. However, if the final hop of the message is a negative hop, the class of the message is not incremented, since a message that has taken its last hop waits for no virtual channels. If H is the maximum hops taken by a message and C is the number of colors used, the maximum number of virtual channels required by the NHOP WH algorithm is

$$1 + \left\lfloor \frac{(C-1)(H-1)}{C} \right\rfloor \quad (2)$$

Proof of Deadlock Freedom. Consider the following partial order on $\boldsymbol{b}$. Given two distinct buffers b, b' in $\boldsymbol{b}$, $b < b'$ if one of the following holds:

1) Class(b) < Class(b'), or
2) Class(b) = Class(b') and Color(b) < Color(b').

Class(b) is the class of b, and Color(b) is the color or label of the node to which b belongs. Now consider a message that takes a hop from buffer b to b'. If the hop is a negative hop, then according to rule 1, b' is greater. Otherwise, according the NHOP, Color(b) is smaller than Color(b'), in which case rule 2 says that b' is greater. Hence, the buffers occupied by any message in successive hops in the SAF routing algorithm have monotonically increasing ranks. Therefore, by Corollary 1 the NHOP wormhole algorithm is deadlock free.

Application to Meshes and Tori. To implement the NHOP wormhole algorithm, we need to demonstrate a suitable coloring scheme. We partition the node set of a (k, n)-torus or (k, n)-mesh network into two subsets: P_0, P_1. The subset to which a node $x = (x_{n-1}, \ldots, x_0)$ belongs is determined using the following rule:

$$x \in P_0 \text{ if } \left(\sum_{i=0}^{n-1} x_i\right) \bmod 2 = 0, \text{ or } x \in P_1 \text{ otherwise.}$$

For even k, the underlying graph of the (k, n)-torus is bipartite, and the partitioning colors the graph. Because adjacent nodes are in distinct subsets, a message takes alternating positive and negative hops along its path from the source to the destination. Therefore, the maximum number of negative hops in a (k, n)-torus with even k is $\lceil n\lfloor k/2 \rfloor/2 \rceil$.

For odd k, the (k, n)-torus is not a bipartite graph and the partitioning does not color the graph. The adjacent nodes connected by wraparound links belong to the same subset (and have the same color), and thus do not meet the criterion of the NHOP routing method; for example, nodes (0, ..., 0, 0) and (0, ..., 0, $k-1$) have the same color if k is odd. (Any pair of adjacent nodes that are not connected by wraparound links will be in distinct subsets and, hence, do not pose a problem.) To solve this problem, assume that for every pair of nodes a and b connected by a wraparound link, there is an imaginary node c between a and b on the wraparound link; further, assume that this imaginary node belongs to the subset other than that of a and b. Thus a hop on the wraparound link from node a to b passes from a to the imaginary node c and then from c to b. One of these hops is a negative hop. The net effect is to increase the maximum number of hops (for counting negative hops only, the actual routing is still minimal) in a dimension by 1, to $\lceil k/2 \rceil$, for odd k.

In summary, a (k, n)-torus has $n\lceil k/2 \rceil$ hops. Since the graph of a (k, n)-mesh is bipartite, for both odd and even k, the total hops is $n(k-1)$. Using $C = 2$ and substituting for H, depending on the type of network, in (2), we obtain that the number of virtual channels needed is at most $1 + \lfloor n\lceil k/2 \rceil/2 \rfloor$, for a ($k$, n)-torus, and $1 + \lfloor n(k-1)/2 \rfloor$ for a (k, n)-mesh.

```
Algorithm Negative-Hop
(Initially, current-class = 0 and current-host
    = source of the message.)
  If (current-host ≠ destination) then {
1) If color of the current-host is 0 or colors of
   previous-host and current-host match, then in-
   crement current-class by one.
2) Select any neighbor node that is in a shortest
   path to destination as the next-host.
3) Reserve the virtual channel of class current-
   class.
4) If the virtual channel is available, set previ-
   ous-host ← current-host, current-host ← next-
   host, and route the message; otherwise, go to
   step 2.
  }
  Else Consume the message
```

Fig. 3. Pseudocode to process a message by the negative-hop wormhole routing algorithm in (k, n)-mesh and (k, n)-torus networks.

When a message is generated, the total number of negative hops taken is set to zero, and the current host is set to the source node. The pseudocode in Fig. 3 describes how a message is routed as per the negative-hop scheme. A message, when it moves from a node of color 0 to a node of color 1, reserves a virtual channel of the same class it reserved in the previous hop; otherwise, it reserves a virtual channel one class higher than what it reserved in the previous hop. The class of a message is also incremented if it takes a hop between nodes of the same color. For the parti-

tion we have described, this can happen only for hops on wraparound links in odd radix (k, n)-torus.

The NHOP is illustrated in Fig. 4 for a message from (2, 2) to (0, 0) in a 4×4 mesh using four virtual channels. The second and fourth hops are negative hops, but the message class is incremented after the second hop only.

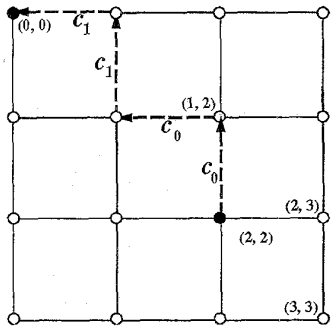


Fig. 4. Example of the negative-hop routing in a 4×4 mesh.

3.2 Improved Hop Schemes

For many networks, the NHOP may require too many virtual channels. The channel requirements can be reduced using improved negative hop schemes (INHOPs), which are based on the negative hop scheme. The basic technique given by Gopal [20] is as follows.

The SAF Version. The network is partitioned such that there are no cycles in any partition, and each partition is given a unique number. Now a negative hop is a hop that takes a message from a node in a higher numbered partition to a node in a lower numbered partition. The hops between nodes in a partition and hops from a lower numbered partition to a higher numbered partition are nonnegative hops. Gopal [20] proves that if H_N is the maximum number of negative hops taken by any message under the improved negative-hop scheme, then $H_N + 2$ buffers are enough for deadlock-free routing. One of these $H_N + 2$ buffers is required to handle direct deadlocks that exist when messages between neighbors in the partition are exchanged. (Direct deadlocks do not occur in the original negative-hop scheme, as per which any pair of adjacent nodes are in distinct partitions.)

The WH Version. A message can use any hop that takes it closer to its destination. A message that has taken i negative hops uses a c_i virtual channel for its next hop. Direct deadlocks cannot occur with wormhole switching, since messages exchanged between neighbors use distinct physical channels. Direct deadlocks occur with SAF switching because of the centralized buffer pool. Therefore, the INHOP wormhole algorithm requires at most

$$1 + \left\lceil \frac{H_I(C-1)}{C} \right\rceil \tag{3}$$

virtual channels, where H_I is the maximum number of inter-partition hops a message can take and C is the number of distinct partitions. It is noteworthy that we use H_I not $H_I - 1$ as in (2), since a message that has taken its final inter-partition hop may still use virtual channels within a partition.

Proof of Deadlock Freedom. The store-and-forward INHOP is free of indirect deadlocks and minimal—a message never revisits a node. Therefore, from Lemma 2, the wormhole algorithms derived from the INHOP are deadlock free.

Application to Meshes and Tori. Compared to the NHOP scheme, the INHOP reduces the buffer requirements for SAF routing by approximately a factor of $n/(n-1)$. First, we apply the INHOP scheme to meshes. The nodes of a (k, n)-mesh are partitioned into two subsets: P_0, P_1. The subset to which a node $x = (x_{n-1}, \ldots, x_0)$ belongs is determined using the following rule: $x \in P_0$ if $\left(\sum_{i=1}^{n-1} x_i\right) \bmod 2 = 0$, or $x \in P_1$ otherwise.

Given any two distinct nodes x, y that belong to the same subset, there is a single path between x and y within the partition if differ only in the DIM_0 component of their addresses, or there is no path between them without involving inter-partition hops. Therefore, there are no cycles in any partition. In fact, the proposed partitioning is equivalent to bipartite coloring of an $(n-1)$-dimensional mesh, and a k-ary n-dimensional mesh is the graph product of a $(k, n-1)$-mesh and a k-node linear chain [23]. Since a message remains in the same partition as long as it takes hops in DIM_0 (row in a 2D mesh) and moves from one partition to another when it takes a hop in DIM_i, $i > 0$, the maximum number of inter-partition hops a message can take is $(n-1)(k-1)$. Hence, the maximum number of negative hops is $\lceil (n-1)(k-1)/2 \rceil$.

Similar reductions in the number of buffers can be obtained for tori also. However, partitions now contain cycles due to wraparound links in DIM_0 For odd k, wraparound connections in other dimensions also cause problems. Both can be solved by treating hops on wraparound connections as negative hops, appropriately. The argument used for the NHOP on odd radix tori applies here with suitable modifications. The maximum number of negative hops for a (k, n)-torus is $\lceil (n-1) \lceil k/2 \rceil / 2 \rceil + 1$.

Therefore, the virtual channel requirements are

$$(k, n)\text{-mesh:} \quad \lceil (n-1)(k-1)/2 \rceil + 1, \tag{4}$$

$$(k, n)\text{-torus:} \quad \lceil (n-1) \lceil k/2 \rceil / 2 \rceil + 2. \tag{5}$$

For a $16 \times 16 \times 16$ torus, 10 virtual channels per physical channel are sufficient and, for a $16 \times 16 \times 16$ mesh, 16 virtual channels are sufficient.

3.3 Negative Hop Scheme Based on Coloring Links

The negative hop scheme above is based on the concept of coloring nodes such that any cycle in the network involves nodes of more than one color. This concept can be naturally extended to coloring links rather than nodes. The edges of the underlying graph of the network are colored such that any cycle involves edges of two or more colors. The two physical channels (one in each direction) of a link are given the color of the corresponding edge in the graph.

Consider the following routing scheme. Any hop that takes a message closer to its destination can be used at any time. A just injected message has 0 negative hops. The first

hop of a message is always a nonnegative hop. A negative hop occurs if a message uses a physical channel of color C' after using a physical channel of color C'' and $C' < C''$. A message with i negative hops (including the current hop) will use a virtual channel of class i for its next hop.

LEMMA 3. *Let H be the maximum number of hops a message takes in the routing scheme based on coloring channels and C be the number of colors used. Then,*

1) *the maximum number of negative hops a message takes is given by (6) and*
2) *fully-adaptive deadlock-free wormhole routing can be provided by using the number of virtual channels given by (7).*

$$\left\lceil \frac{C-1}{C}(H-1) \right\rceil \tag{6}$$

$$1+\left\lceil \frac{C-1}{C}(H-1) \right\rceil \tag{7}$$

PROOF. Since the first hop is always nonnegative, at most $H-1$ hops can cause negative hops. Substituting $H-1$ for the number of hops in (1) yields (6). Traveling on links of a color is the same as traveling in a cluster in the INHOP scheme, and each hop, after the first hop, can be on a link of color different from that of the previous one. Therefore, substituting $H-1$ for H_I in (3) yields the upper bound on the number of virtual channels given by (7).

Since there is no equivalent SAF algorithm for this algorithm, we present a direct proof of deadlock freedom by showing that the channel graph of the algorithm is acyclic.

We form one subgraph for each color from the underlying graph of the network with edges colored. The subgraph for color i consists of all the edges of color i and the nodes connected to these edges. Since the coloring is such that cycles cannot be formed with edges of one color only, each of these subgraphs is acyclic.

Let $c1$ and $c2$ be two virtual channels such that $c1$ is an in channel to a node and $c2$ is an out channel from the same node. Let $p1$ and $p2$ be the physical channels of $c1$ and $c2$, respectively. Then $c1 < c2$ if one of the following is true:

1) Class of $c1$ < class of $c2$,
2) Channels $c1$ and $c2$ have the same class, but color of $p1$ < color of $p2$, or
3) Channels $c1$ and $c2$ have the same class, and $p1$ and $p2$ have the same color.

The first two rules are similar to the ones seen for the original NHOP algorithm. Since the algorithm uses shortest paths and since the subgraph of a color is acyclic, there cannot be a cycle within a partition involving $c1$ and $c2$, if $c1$ and $c2$ are ranked using the third rule. So, the ranking of a pair of virtual channels, if specified, by these rules is consistent.

Now, consider a message that uses or waits for $c2$ after acquiring $c1$. If $p1$ and $p2$ are of different colors, then one of the first two conditions above holds, and $c2$ is of higher rank than $c1$. Otherwise, $p1$ and $p2$ are in the same subgraph, and the third condition specifies that $c1 < c2$. Therefore, each message acquires virtual channels of strictly increasing ranks. So, the channel graph is acyclic. □

Application to Meshes and Tori. First consider a (k, n)-mesh, since it presents the simpler case. Channels in DIM_i, $0 \le i < n$, are given color i. For example, in a 2D mesh, all row (DIM_0) channels are of color 0 and all column channels are of color 1. (Dally and Aoki [12] have presented this method for meshes. But they did not provide any bounds on virtual channels required.) A row hop following a column hop is a negative hop. Thus, the maximum number negative hops is

$$\left\lceil \frac{n-1}{n}\left[n(k-1)-1\right] \right\rceil = (n-1)(k-1).$$

For a (k, n)-torus, we start by coloring channels of DIM_i with color i. Because of the wraparound connections, the underlying graph of a torus has cycles consisting of edges of the same color. To break these cycles, all hops on wraparound links are taken to be negative hops. Then the number of negative hops in a torus can be derived as follows. At most $n(\lfloor k/2 \rfloor - 1)$ hops are taken on grid (non-wraparound) links and n hops on wraparound links. Noting that at most n colors are used for grid links and each wraparound hop is a negative hop, the number of negative hops in a torus is no more than

$$\left\lceil \frac{n-1}{n}\left[n\left(\lfloor k/2 \rfloor - 1\right)-1\right] \right\rceil + n = (n-1)\lfloor k/2 \rfloor + 1.$$

The upper bound on the number of virtual channels required is at most one more than the number of negative hops. The above analysis indicates that this method requires more virtual channels than the NHOP for three and higher dimensional meshes and tori.

3.4 Hop Schemes With Class Upgrades

The hop schemes described thus far do not utilize virtual channels evenly: virtual channels with lower numbers are utilized more than virtual channels with higher numbers. For example, all messages use virtual channels of class 0, but only messages between diametrically opposite nodes (very few) use virtual channels in the highest numbered class. A slight modification to any of the three routing algorithms corrects this situation and achieves a more uniform utilization of virtual channels.

We discuss this modification for the NHOP scheme. The modified scheme is called negative-hop with class upgrades. The modification is to give each message a few *bonus upgrades* based on the number of negative hops it can take before reaching its destination. The number of bonus upgrades a message M receives at its source node is given by the following formula.

$$\begin{aligned}&\text{Number of bonus upgrades} =\\ &\quad \text{maximum number of negative hops possible}\\ &\quad - \text{number of negative hops to be taken by M}\end{aligned} \tag{8}$$

A message with no bonus-upgrades is routed exactly the same as in the NHOP algorithm. A message with b bonus-upgrades, $b \ge 0$, may start its journey using a virtual channel in one of $c_0, \ldots c_b$ classes; the remainder of its journey is governed by the NHOP algorithm given in Fig. 3. This is called the static bonus upgrades method. In the dynamic bonus upgrades method, a message may keep its bonus

upgrades and, at any time during the journey, upgrade its virtual channel class by expending one or more bonus upgrades. The dynamic class upgrades method is more expensive to implement, and our experience indicates that both dynamic and static methods have similar performances.

Since a message never waits for a lower class virtual channel, even with class upgrades, the routing is deadlock free. In addition to balancing the load on virtual channels, this method gives priority to messages traveling short distances, which improves performance, especially for highly local traffic [6].

3.5 Hop Schemes With Class Ranges

Another improvement we can incorporate into hop schemes is to give more choice of virtual channels for messages in higher classes. For example, a message with virtual channel class $i \geq 0$ may use any virtual channel of classes $0, \ldots, i$. The actual implementation is as follows. If a message of class 2 does not find a virtual channel of class 2 in the path to its next host, the message selects any free virtual channel in classes 0 and 1 that is in its path, relabels it as 2 and uses it. A virtual channel relabeled by a message of higher class number returns to its original class after the message has relinquished it. A blocked message, however, can only wait for a virtual channel of its class.

Deadlocks cannot occur, since each blocked message waits for virtual channels as per the original algorithm. Starvation may be avoided by ensuring that a virtual channel is relabeled to a higher class only when there are no messages of its class waiting for it. Using *ranges* of classes to select virtual channels gives priority to messages that have already used many virtual channels. The use of both class upgrades and ranges has the undesirable effect of giving low priority to messages that need to travel long distances and, perhaps, should be avoided.

4 Wormhole Routing Algorithms for de Brujin and n-Star Networks

Our design techniques are not limited to k-ary n-cube networks. To illustrate this, we design new wormhole algorithms using the theory developed thus far for multicomputer networks based on de Brujin [29] and n-star graphs [2].

4.1 Wormhole Routing in de Brujin Networks

A k-ary n-dimensional de Brujin network has k^n nodes. In this paper, we consider only binary de Brujin (or, simply, de Brujin) networks, but our results can be extended to radix-k de Brujin networks easily.

First, we consider de Brujin networks with unidirectional links whose underlying graphs are directed de Brujin graphs. An example of directed 3D de Brujin network is shown in Fig. 5. In general, a binary nD de Brujin network has diameter n, and the in and out degrees of a node is 2. In particular, node $x = x_{n-1} \ldots x_0$ is connected to the following two nodes:

$$\sigma_0(x) = x_{n-2} \ldots x_0\, 0 \text{ and } \sigma_1(x) = x_{n-2} \ldots x_0 1.$$

The connections out of a node are called $\sigma_{0,1}$—leftshifts with 0 or 1 fill—connections. Nodes 0 and $N-1 = 2^n - 1$ are exceptions in that one of their edges results in a loop. For the sake of clarity we ignore the loops. When the directions of all edges are reversed, we get yet another type of de Brujin network, which uses σ_{-1} connections—right shifts with 0 or 1 fill.

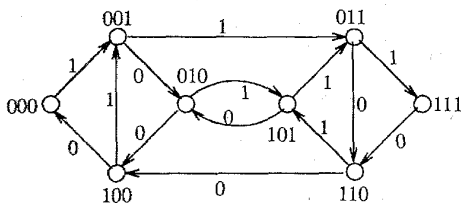


Fig. 5. A directed three dimensional binary de Bruijn network. The loops at nodes 0 and 7 are omitted for clarity. The type of an edge is indicated by a 0 or 1 as appropriate.

There is only one shortest path between any pair of nodes. Hence, with minimal routing there is no adaptivity in a directed de Brujin network. But deadlocks occur if multiple virtual channels are not used. Therefore, we investigate the issue of deadlock-free minimal routing.

Since binary de Brujin graphs are not bipartite for $n \geq 2$, a minimum of three colors are needed to apply the NHOP scheme. Ganesan and Pradhan [18] indicate that three colors are sufficient to color a de Brujin graph. From their result and (2), it is follows that $1 + \lceil 2(n-1)/3 \rceil$ virtual channels are sufficient for deadlock free routing.

Using the concept of coloring links rather than nodes, we can reduce the virtual channel requirement further. First we note that the edges of a de Brujin can be grouped into two classes: 0-edges and 1-edges. A 0-edge connects a node to another node with a 0 in the last bit position, and 1-edge connects a node to another with a 1 in the last bit position (see Fig. 5).

LEMMA 4. *Let $G = (V, E)$ be a binary, directed de Brujin graph with node set V and edge set E. Let E_0 indicate the set of all 0-edges and E_1 the set of all 1-edges. Then,*

a) $E_0 \cup E_1 = E$ and
b) the directed subgraphs $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$ of G are acyclic.

PROOF. Part (a) of the lemma is true by the construction property of the de Brujin graph.

We now prove part (b) for G_0. Assume that G_0 has at least one cycle. That is, there exists a sequence of nodes $x1, x2, \ldots, xm$, $m > 1$, such that $\sigma_0(x1) = x2$; $\sigma_0(x2) = x3$; ...; $\sigma_0(xm) = x1$. Then one of the nodes must be node 0, since n consecutive hops on 0-edges from any node lead to node 0. But the 0-edge of node 0 results in a loop. Therefore, the above cycle has a break after the occurrence of node 0. This is a contradiction, and G_0 is acyclic.

A similar argument can be constructed for G_1. If there is a cycle of 1-edges then it should have node $N - 1 = 11 \ldots 1$. But node $N - 1$ is not connected to any other node with a 1-edge. □

Since there are only two types of edges, we need two

colors: color 0 for 0-edges and color 1 for 1-edges. (For a k-ary de Brujin graph, k colors are used.) There cannot be cycles when edges of only one color are used. The first hop is always nonnegative. A hop on a 0-edge immediately after a hop on a 1-edge is a negative hop. For each nonnegative hop, the current virtual channel class is used. For each negative hop, a message uses a virtual channel of one class higher than the current one. Using (7), we conclude that

$$1+\lceil (n-1)/2 \rceil = \lceil (n+1)/2 \rceil$$

virtual channels are sufficient for deadlock free routing.

An undirected de Brujin network is obtained by replacing the unidirectional links with bidirectional links. A minimal routing algorithm treats an undirected de Brujin network as two directed de Brujin networks: one directed de Brujin has $\sigma_{0,1}$ connections and the other has $\sigma_{0,1}^{-1}$ connections. Lemma 4 holds for both types of directed graphs. With minimal routing, the path of a message lies completely in one of the networks. Therefore, deadlock-free wormhole routing can be provided in undirected de Bruijn networks using the link coloring scheme discussed for directed de Bruijn networks.

Ganesan and Pradhan [18] give a different routing algorithm with $\lceil n/2 \rceil$ virtual channels for binary de Brujin networks. For k-ary de Brujin networks, our algorithm requires $1+\lceil (n-1)\ (k-1)/k \rceil$ virtual channels. Park and Agrawal [37] give a different routing algorithm with similar bounds on virtual channels.

4.2 Wormhole Routing in n-Star Networks

Star graphs belong to the class of Cayley graphs studied by Akers and Krishnamurthy [2]. The number of nodes in an n-degree star graph (or simply n-star) is $n!$ and the degree of a node is $n-1$.

An n-star network has an n-star graph as its underlying graph. It is convenient to associate each node of an n-star with a unique permutation of the integer sequence $1,\ldots, n$. Two nodes of a star graph are connected by an edge if the label of one can be obtained from the other by interchanging the symbol in the first (leftmost) position with the symbol in some other position. The operation of interchanging symbols in positions 1 and i of a permutation is the transposition $(1, i)$ and is denoted t_i. Hence, each edge of a star graph can be labeled with t_i for some $2 \le i \le n$. For example, node 1342 in the 4-star graph in Fig. 6 is connected to nodes 3142, 4312, and 2341 using edges with labels t_2, t_3, t_4, respectively.

To apply the hop schemes, we investigate the chromatic number of the star graph.

LEMMA 5. *The n-dimensional, $n \ge 0$, star graph is bipartite, and hence can be colored with two colors.*

PROOF. We prove the lemma by giving a coloring scheme. Recall that the label of each node in an n-star is a permutation of the identity permutation $I = 12 \ldots n$. The identity permutation I (and its associated node) is given color 0. We give color 0 to a permutation P if and only if P can be obtained by applying an even number of transpositions of the form t_i, $2 \le i \le n$, on I; otherwise, P is given color 1. From a well-known result in the theory of permutations [24], each permutation is assigned a unique color.

To complete the proof, we need to show that adjacent nodes have opposite colors. If two nodes x and y are adjacent, then there exists a transposition t_i, $2 \le i \le n$, such that x is obtained by applying t_i on y. Therefore, if x is of color 0, then y is of color 1, and vice-versa. □

Akers and Krishnamurthy [2] prove that the diameter of a star graph is $\lfloor 3(n-1)/2 \rfloor$. Substituting $C = 2$ and $H = \lfloor 3(n-1)/2 \rfloor$ in (2), we obtain that

$$1+\left\lfloor \frac{\lfloor 3(n-1)/2 \rfloor}{2} \right\rfloor$$

virtual channels per physical channel are sufficient for deadlock-free wormhole routing in an n-star. The previously best known bound on virtual channels is $n-1$ by Misic [33].

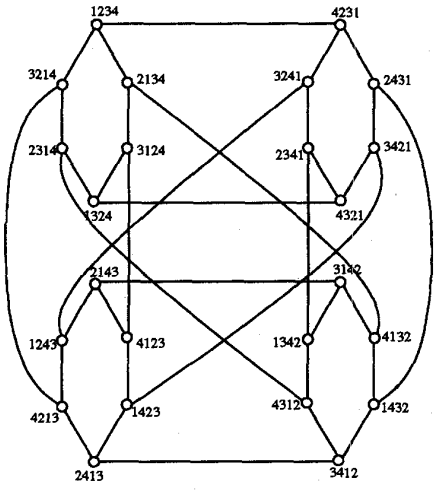


Fig. 6. A 4-star network.

5 Implementation and Performance Considerations

In this section, we investigate the resource requirements and performances of wormhole schemes derived from SAF algorithms, in general, and the NHop scheme, in particular. Since majority of the studies and implementations are specific to mesh and torus networks, we use these networks as examples in our analyses. Since our algorithms are general enough to apply to any network, our delay and cost analysis may be applied in the design of routers for other networks also. We start with a discussion on router organizations.

In normal wormhole routing, each virtual channel has a dedicated flit buffer to hold the flit transmitted on the virtual channel. Therefore, deadlocks on flit buffers is not an issue in wormhole routing. A possible datapath organization of an adaptive router [8] is shown in Fig. 7. But as the degree of a node increases, the buffer requirement for the entire node increases, even when the number of virtual channels per physical channel is constant. This problem is exacerbated when deep buffers (to hold multiple flits and improve latencies) are used.

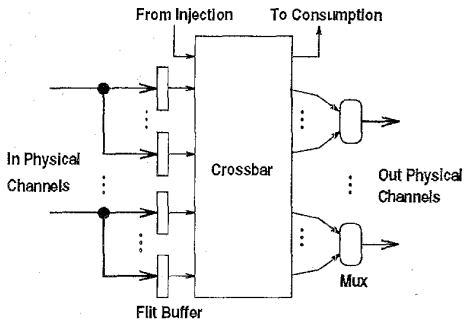


Fig. 7. Datapath of wormhole router with dedicated flit buffers.

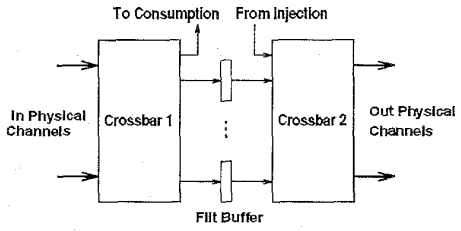


Fig. 8. Datapath of wormhole router with centralized flit buffers.

Therefore, IBM's Vulcan network provides *centralized* flit buffers in each router to improve performance [42]. Each Vulcan switch has a central queue of 1,024 bytes shared by all the eight incoming channels. To ensure deadlock free routing, however, the Vulcan switch provides a dedicated flit buffer on each input channel. An alternative datapath organization with centralized buffers to facilitate sharing of flit buffers among multiple channels is shown in Fig. 8. The WH algorithms that use only dedicated flit buffers can also be implemented using the centralized organization. The main difiference is each virtual channel goes through a crossbar before accessing its exclusive buffer.

5.1 Cost and Delay Analysis of Centralized Buffers Organization

Consider some WH routing algorithm, which requires dedicated flit buffers for each virtual channel, but is implemented using the centralized organization. Flit buffers are not shared; each virtual channel has its exclusive buffer, but needs to go through Crossbar 1 (in Fig. 8) before accessing its exclusive buffer. We will compare the router delay and cost for such an algorithm implemented using dedicated buffers and centralized buffers organizations. We assume that m is the number of flit buffers used, p the number of incoming physical channels to a router, and v the number of virtual channels per physical channel. It is clear that $m = pv$.

Router Delay. For dedicated buffers organization, the major components of delay are flow control from incoming physical channels to flit buffers, crossbar delay from flit buffers to the outputs of the central crossbar, and the virtual channel controller delay from the outputs of the crossbar to outgoing physical channels [8]. For header flit,

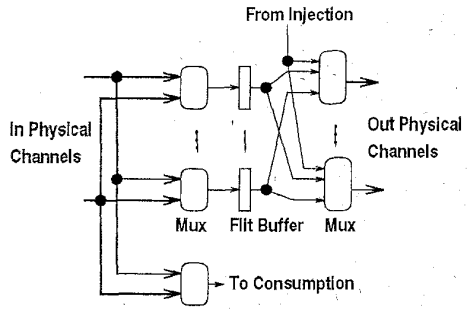


Fig. 9. Logical organization of the wormhole router with centralized flit buffers.

header decode and update and channel selection are the additional costs. We compare various delays of the centralized buffers organization with the dedicated buffers organization.

The header decode and update and channel selection are similar for both organizations. The flow control in the centralized organization is done in Crossbar 1. So, when a header flit arrives, say, from NODE_1 (node 1) to NODE_2, it is allocated a central buffer by establishing a connection through Crossbar 1 of NODE_2 or is refused connection. The header is retained by NODE_1 for a few cycles, by which time rejection of the header, if occurred, will be known. Once the connection is established, the allocated central flit buffer acts as a dedicated flit buffer to that virtual channel, and the transit of data flits is similar to that of dedicated flit buffer implementation. The crossbar delay in the dedicated buffers organization is eliminated in the centralized buffers organization. The virtual channel controller is implemented using Crossbar 2.

The centralized buffer organization in Fig. 8 may indicate that Crossbar 2 of a node and Crossbar 1 of the next node must be switched in a coordinated fashion to transmit a flit between the two nodes. This is not true, however. To show this, we give the logical organization of the centralized router in Fig. 9. Comparing Fig. 8 and Fig. 9, we notice that the operation of the first column of multiplexers in the logical organization is implemented by Crossbar 1, and that of the second column of multiplexers by Crossbar 2.

Once a flit buffer is allocated to a virtual channel, it remains associated with that virtual channel until it is released. Therefore, a multiplexer between the inputs and buffers in the logical organization is set once at the time of setting up the path. An input channel may be allocated multiple flit buffers, one for each active virtual channel on the input channel. Since a crossbar naturally provides the multicast communication, this can be accomplished easily by setting one input channel to flit buffer connection for each request accepted and removing one such connection for each request completed. Flits coming on a physical channel are available at all the flit buffers allocated to it and an appropriate flit buffer accepts the flit. This is similar to storing a flit in one of the appropriate flit buffers associated with the physical channel in the dedicated buffers organization.

The amount of switching done by Crossbar 2 is the same as the amount of switching done by the multiplexers at the

output physical channels in Fig. 7. This crossbar changes its settings on flit-by-flit basis, much the same way the multiplexers in Fig. 7 change their settings.

If the flit size is such that it takes multiple cycles to transmit between nodes (for example, in Cray T3D, it takes six to eight cycles to transmit a flit from one node to another), then Crossbar2 will have ample time to change settings.

In summary, connection from an input virtual channel to an output virtual channel takes more time, and data flits go through two smaller crossbars instead of one large crossbar with centralized organization. But the centralized organization with buffers between the crossbars lends itself easily to pipelining, there by avoiding increase in clock cycle time. Since the centralized organization has longer datapath, the router delay for a message increases compared to the dedicated organization, when the number of buffers is kept the same.

Router Cost. Since the number of buffers is kept constant, there are two cost components: number of crosspoints for the crossbars, and the wire area for multiplexers and the rest of the data path. Let w be the flit size.

Even with the simplest hierarchical implementation, the multiplexers require $O(w^2\, m \log_2 v)$—wm wires horizontally and $\log_2 v$ levels with w wires vertically, at each level. So, crosspoint area, which is approximately w^2 times the number of crosspoints, dominates the overall silicon area for both routers.

The number of crosspoints used, for the dedicated buffer implementation, is $(m + 1) \times (m + 1) \simeq m^2$ and, for the centralized buffer implementation, $2(p \times (m + 1)) \simeq 2pm$. If $m \geq 2p$, then the cost of the centralized buffers organization is comparable to that of the dedicated buffers organization.

5.2 Buffers Requirements of Wormhole Algorithms

We are now ready to compare the resource requirements of more traditional WH algorithms with those designed from SAF algorithms. From the above discussion, if the total number of buffers used is the same, then a wormhole scheme that has high virtual channel requirements but does not require dedicated flit buffers remains competitive with more traditional wormhole algorithms that require typically a constant number of virtual channels with dedicated flit buffers. So, the critical issue is the buffer requirements for SAF based wormhole algorithms.

We show that buffer requirements can be substantially reduced for wormhole algorithms derived from certain SAF algorithms. The idea is to provide m classes of centralized flit buffers in WH routing if m classes of packet buffers are used in SAF routing. No dedicated buffers are provided for individual virtual channels. A head flit, on arriving at a node, will use a flit buffer of class i, where i is the class of the packet buffer that will be used for this message in SAF routing.

LEMMA 6. *If a store-and-forward algorithm ensures that messages acquire buffers in the greater than order, $\succ$, of some partial order on $\mathbf{b}$, the corresponding wormhole algorithm is deadlock free even when only centralized, and no dedicated, flit buffers are provided.*

PROOF. With centralized flit buffers, additional dependencies occur on flit buffers. To handle this, we consider the expanded resource graph of the WH algorithm in which the resources are virtual channels and centralized flit buffers. We start by giving a ranking of virtual channels and centralized flit buffers. Let b be a centralized flit buffer in node, say, x. If c is an input virtual channel to node x such that a message arriving into node x through c can use b, then $c < b$. Similarly, if c is an output virtual channel to node x such that a message using buffer b can leave x using c, then $b < c$. This gives a ranking of centralized flit buffers which is also the ranking of packet buffers in the underlying SAF algorithm. Therefore, there cannot be cycles involving two or more centralized flit buffers in the resource graph of the WH algorithm. □

COROLLARY 2. *The wormhole NHOP algorithm is deadlock free even when only m flit buffers are provided per node, where m is the number of virtual channels given by* (2).

PROOF. We have shown in Section 3.1 that the SAF version of NHOP satisfies the hypothesis of Lemma 6. Therefore, the wormhole NHOP is deadlock free when the m flit buffers are organized as m classes of centralized flit buffers. □

It is easy to show that the above corollary holds for NHOP with class ranges and upgrades as well.

For many known WH algorithms, the centralized buffers organization does not reduce buffer requirements. For example, the e-cube requires two classes of virtual channels. But providing two centralized buffers–one buffer for each virtual channel class—does not work, since direct deadlocks occur. For such algorithms, virtual channels that are used to prevent deadlocks should have exclusive flit buffers (with either router implementation) to avoid deadlocks.

With dedicated buffers implementation, the NHOP scheme requires too much memory per router. With centralized flit buffers, however, it requires less memory. To see the implications of Lemma 6, let us consider the e-cube algorithm, which requires $4n$ dedicated flit buffers for a (k, n)-torus, for example, 12 for a 3D torus. With centralized buffers, the NHOP requires $1 + \lfloor n\lceil k/2 \rceil / 2 \rfloor$ buffers, which is seven for an (8, 3)-torus. In fact, the NHOP requires fewer buffers than the e-cube for (k, n)-tori with $k \leq 14$. For $k = 15$, 16, the NHOP requires one more buffer than the e-cube. Similarly, for a (k, n)-mesh with $k \leq 5$, the NHOP requires fewer or just one more buffer than the e-cube.

Some of the recently proposed fully-adaptive algorithms for k-ary n-cube networks require only a constant number of virtual channels. Two recent examples of such fully-adaptive algorithms are the *-channel algorithm [4], [14] for tori and the Opt-Y algorithm [40] for meshes. The *-channel scheme requires three classes of virtual channels and is based on the e-cube algorithm: two virtual channel classes are used to avoid deadlocks and the extra class is used to provide adaptive (non e-cube) routing. The Opt-Y algorithm requires only two virtual channels: one class is used to avoid deadlocks using the West-First algorithm [19] and the other class to provide full-adaptivity.

With dedicated flit buffers, the *-channel scheme requires a minimum of $6n$ buffers for an nD torus (18 for a 3D

torus). It is possible to reduce the requirement, by providing dedicated flit buffers for the *e*-cube channels (to preserve deadlock freedom) and centralized flit buffers for adaptive channels. With as few as $4n + 1$ buffers, fully-adaptive routing can be provided by the *-channel scheme. (Care should be taken here to avoid deadlocks, since the number of adaptive channels is more than the number of buffers available for messages on the adaptive channels, and obtaining an adaptive virtual channel does not guarantee a flit buffer in the node at the other end of the channel. In particular, a message should not attempt to use an adaptive channel, when *e*-cube channel is available.) Still, when $k \leq 16$, the NHOP requires fewer or the same number of buffers per router. The Opt-Y scheme requires only two virtual channels. So, its buffer requirements grow as $4n$ with all dedicated flit buffers or $2n + 1$ with buffers for the adaptive channels shared. Since a (k, n)-mesh has almost twice the diameter of a (k, n)-torus, the NHOP requires more buffers. For example, unless, $k < 7$, the NHOP requires more buffers than the Opt-Y scheme in a (k, n)-mesh.

The cost comparisons indicate that the NHOP with centralized organization of routers could be an attractive alternative for tori, but less attractive for large-radix meshes.

5.3 Performance Comparisons

We have used a register-transfer level simulator to compare the performances of the NHOP, *e*-cube, *-channel, and Opt-Y schemes. For the NHOP algorithm, we used class ranges (see Section 3.5). We have simulated (16, 2)-torus, (8, 3)-torus, and (8, 3)-mesh networks with uniform and bit reversal traffic. Uniform traffic is widely used in simulation studies and serves as a benchmark traffic pattern. The bit reversal traffic creates multiple hotspots and severely tests the adaptivity of an algorithm. In practice, fixed length messages give better manageability of resources such as injection and consumption buffers, and small message sizes are more suitable for fine-grain computations. Hence, we have used fixed length messages of 20 flits, which could be suitable for transmitting four 64-bit words together with header, checksum and other information on 16-bits wide physical channels such as the ones used in Cray T3D.

We have assumed a centralized organization for NHOP routers, and dedicated organizations for *e*-cube, Opt-Y, and *-channel routers. Based on the above discussion on router delays, we have assumed that the NHOP router takes three cycles to set up an appropriate connection for an incoming message; if the connection is already set up, then data flits have two cycles latency through the router. On the other hand, delay through the router node was uniformly set to one cycle for *e*-cube, Opt-Y, and *-channel schemes, to reward their use of fewer virtual channels and dedicated buffers. The clock cycle time is the same for all routers. In each cycle, a router, if it has one or more message headers waiting for connections, attempts to set up connection for one header, selected in round-robin manner, by checking the virtual channels specified by its algorithm.

To facilitate simulations at and beyond saturation, we have used a congestion control mechanism: a node is not permitted to inject new messages into the network if a certain number of its previously injected messages are still within its router. This number, estimated using some preliminary simulation runs, is between six and eight (depending on the network simulated) for uniform traffic and between three and six for bit reversal traffic. This mechanism has no effect on the router delay and throughput prior to saturation, and helps sustain network throughput for traffic rates beyond saturations. Despite this congestion control, sometimes the curves double back indicating that peak throughputs in such cases are not sustained for traffic rates beyond saturation.

In wormhole routing, bubbles could be introduced, especially at low traffic, in the transmission of consecutive flits of a message because of asynchronous pipelining. To reduce these bubbles, we used buffers of depth 4, that is, each buffer can hold four flits of a single message. Whenever, a buffer has space for two or more flits, next pair of data flits are sent from the previous router in the path. All other parameters are kept the same in all simulations.

We have used average time spent in the network by a message and the utilization of bisection bandwidth as the performance metrics. For all the results in this paper, 95%-confidence intervals [26] are ±5% of the respective values reported. All the graphs show message latency in cycles versus achieved bisection utilization.

Torus Simulations. We have simulated two cases for an (8, 3)-torus: 18 or 24 buffers per node. Fig. 10 and Fig. 11 show the performances of the *e*-cube, NHOP and *-channel schemes for the 18 buffers case. All algorithms are given 18 buffers per node, which is the minimum number required by the *-channel

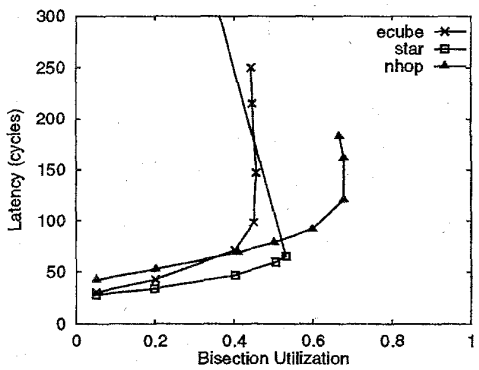


Fig. 10. Performance of the *e*-cube, *-channel, and NHOP algorithms for uniform traffic in an (8, 3)-torus with 18 buffers per node.

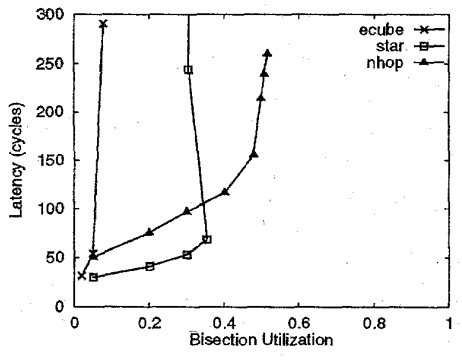


Fig. 11. Performance of the *e*-cube, *-channel, and NHOP algorithms for bit reversal traffic in an (18, 3)-torus with 18 buffers per node.

scheme. For the *e*-cube, we simulated, on each physical channel, three virtual channels, two for deadlock free routing and one usable by either class depending on traffic. For the NHOP, we have simulated seven (the minimum required) virtual channels with centralized buffers organization. For the *-channel scheme, we have simulated three virtual channels, two for deadlock free *e*-cube routing and one for adaptive routing. We have used Duato's channel selection policy, as per which adaptive channels are used whenever they are available [14]. (We have compared and found that our results on the *-channel algorithm are consistent with the simulation results of Duato and Lopez [15] for an algorithm closely related to the *-channel scheme.)

From Fig. 10 and Fig. 11, it is clear that the NHOP has higher latency at low traffic, because of longer router delays, but offers higher throughput—26% higher for uniform and 46% higher for bit reversal—than the *-channel scheme.

We have simulated only the *-channel and NHOP algorithms for the 24 buffers case. For the *-channel, we have provided two channels for adaptive routing and two for deadlock free routing. For the NHOP, we have simulated eight virtual channels (the eighth channel is shared by all classes of messages) and centralized buffers organization. Fig. 12 and Fig. 13 give the performances of the two algorithms for uniform and bit reversal traffics. Once again, the NHOP has higher latency at low traffic but offers higher throughput. (To see if the *-channel scheme offers higher performance, we have simulated it with centralized buffers for adaptive channels, and with the number of adaptive channels varied from two to five. The peak throughput remained the same or reduced.)

We have also simulated the *e*-cube, *-channel and NHOP for a (16, 2)-torus with 16 flit buffers per node. The results are given in Fig. 16 and Fig. 17. For the 2D torus, the NHOP offers higher throughput but the *-channel scheme has similar performance with lower latency.

Mesh Simulations. Since NHOP requires more buffers than *e*-cube and Opt-Y algorithms, we have simulated only an (8, 3)-mesh with 24 buffers per node. The results are given in Fig. 16 and Fig. 17. In this instance, the Opt-Y outperforms the NHOP.

5.4 Summary of Cost and Performance Comparisons

The cost of a wormhole router is often associated with the number of virtual channels simulated on each physical channel. This is an appropriate measure of cost when dedicated flit buffers and the router organization in Fig. 7 are used. When flit buffers are shared among multiple virtual channels, and centralized buffers organization of Fig. 8 is used, the total number of flit buffers, not the virtual channels, determines the router cost.

Thus, even with wormhole routing, buffer area is a major limiting factor in designing router chips. With centralized buffers, and appropriately designed routing algorithms, it is feasible to provide fully-adaptive routing using less buffer area than that required for *e*-cube or other traditional WH algorithms. This is especially true for tori, for which the NHOP requires fewer buffers than the *e*-cube for $k \leq 14$. The NHOP is not as attractive for meshes because of large diameters and reduced virtual channel requirements for the *e*-cube and fully-adaptive schemes such as Opt-Y.

Our simulation study indicates that an NHOP based algorithm gives higher throughput than the *e*-cube and *-channel schemes for both uniform and bit reversal traffic in tori. The NHOP performs worse than the Opt-Y scheme for meshes, however, probably because of network asymmetry and high diameter.

Chien [8] showed that, for nonpipelined routers, using many virtual channels incurs high costs and longer clock cycle times compared to, for example, the *e*-cube. So, an adaptive router may have more flits delivered per cycle than an *e*-cube router, but may have 1.5 or two times longer clock cycle time, resulting in lower throughput in flits per second. In this study, we have considered pipelined routers and have shown that with an appropriate routing algorithm, sharing of flit buffers, and pipelining, both cost and clock rate limitations can be overcome. In the context of pipelined routers, the net effect of a more complex routing algorithm is higher message latencies at low traffic.

6 Concluding Remarks

We have presented a technique to design wormhole algorithms from store-and-forward algorithms. In addition, we have provided a sufficient condition under which the wormhole algorithms are deadlock free. As an application of this technique, we have designed wormhole algorithms from store-and-forward hop schemes known in the computer networks literature [20]. In particular, we have presented the negative-hop (NHop) wormhole algorithm and several variations of it. We have also shown that the previously proposed dimension reversal scheme [12] is a variant of the NHop. Our results are not limited *k*-ary *n*-cubes. To illustrate this, we have given deadlock-free, fully-adaptive wormhole algorithms for de Brujin and *n*-star networks, with the best bounds on virtual channels.

We have considered a new router organization based on the concept of sharing flit buffers by placing them in a central pool. We have shown that if an SAF algorithm routes messages such that buffers are acquired in a monotonically increasing order, then the buffers required for the corresponding wormhole algorithm is reduced by the factor of the node degree. With centralized buffers implementation, it is the total number of buffers used, not the number of virtual channels simulated on each physical channel, that determines cost of the router. To handle longer datapaths and more complex control associated with fully adaptive routers, we have considered pipelining within routers. Pipelining avoids increase in clock cycle time with no significant loss in the number of flits delivered per cycle.

With centralized buffers organization, the NHOP provides, for many configurations of *k*-ary *n*-cube networks, fully-adaptive routing while requiring fewer buffers than the *e*-cube. For example, for the (8, 3)-torus used in a 512 node Cray T3D, the NHOP requires seven flit buffers for fully-adaptive routing, while the *e*-cube requires 12 flit buffers. For the $8 \times 16 \times 8$ torus (the maximum configuration for Cray T3D), the NHOP requires nine buffers, while the *e*-cube requires 12 buffers. Because of longer diameters and simpler routing with *e*-cube in meshes, the NHOP requires more buffers than the *e*-cube, unless $k \leq 5$.

We have evaluated the performances of the NHOP with class ranges, *e*-cube, and previously proposed fully-adaptive *-channel and Opt-Y schemes. Our simulations indicate that

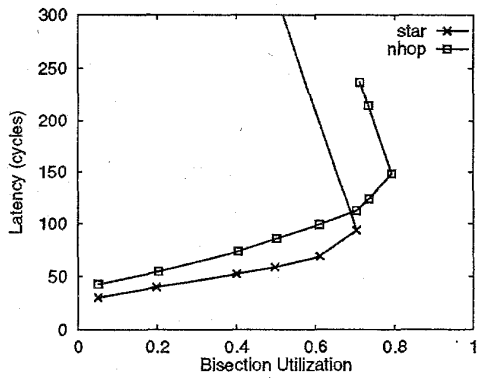


Fig. 12. Performance of the *-channel and Nhop algorithms for uniform traffic in an (8, 3)-torus with 24 buffers per node.

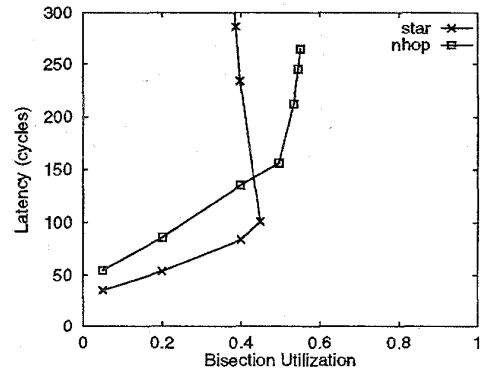


Fig. 13. Performance of the *e*-cube, *-channel and NHop algorithms for bit reversal traffic in an (8, 3)-mesh with 24 buffers per node.

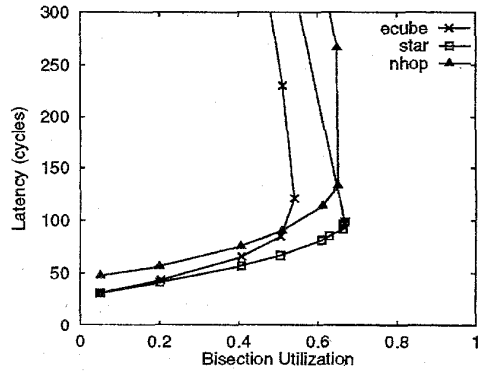


Fig. 14. Performance of the *e*-cube, *-channel, and Nhop algorithms for uniform traffic in a (16, 2)-torus with 16 buffers per node.

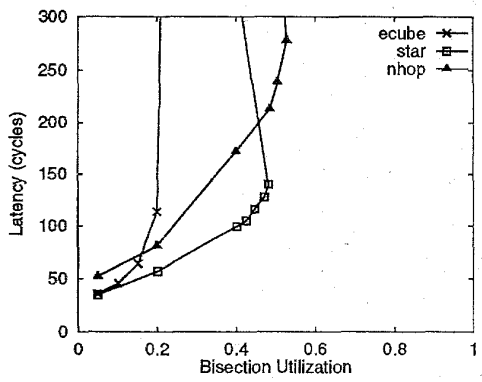


Fig. 15. Performance of the *e*-cube, *-channel, and NHop algorithms for bit reversal traffic in a (16, 2)-torus with 16 buffers per node.

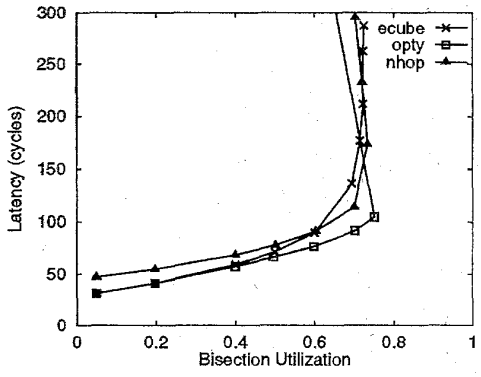


Fig. 16. Performance of the *e*-cube, Opt-Y, and NHop algorithms for uniform traffic in an (8, 3)-mesh with 24 buffers per node.

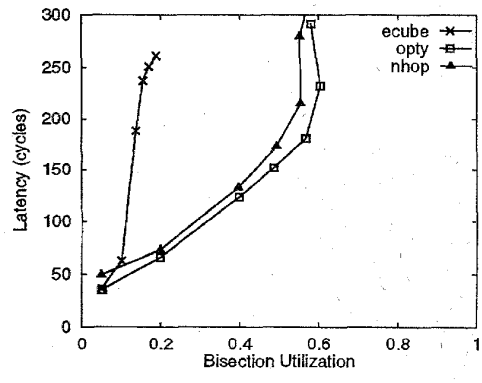


Fig. 17. Performance of the *e*-cube, Opt-Y, and NHop algorithms for bit reversal traffic in an (8, 3)-mesh with 24 buffers per node.

the NHOP performs better than the *e*-cube and *-channel schemes for tori. For meshes, the NHOP performs better than the *e*-cube but slightly worse than the Opt-Y scheme.

Based on the buffer cost and throughput evaluations, the NHOP has advantage 8 over previously proposed wormhole algorithms for torus networks. The NHOP has higher message latency, however, due to centralized router organization.

For the current wormhole algorithms, the buffer requirements increase with node degree or the diameter. A better scalable routing method should use only a constant number of buffers independent of the node degree or network diameter. Such algorithms exist for packet routing on torus and mesh networks [38], [10], but the buffer size increases with packet size. Our results on centralized buffers organization and the sufficient condition for deadlock-free sharing of flit buffers may be used to explore if such algorithms exist for wormhole routing. Similar results for wormhole routing facilitate design of fully-adaptive router modules, from which routers for networks of arbitrary size and node degree can be designed.

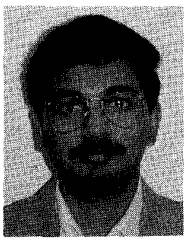
ACKNOWLEDGMENTS

We would like to thank Profs. C.S. Raghavendra and D.K. Panda for many discussions and comments on an earlier draft of this paper, Prof. Ram C. Tripathi for discussions on the convergence criteria, and Mr. Jeff Seigel for developing the simulator. We also thank the anonymous reviewers and Prof. L.M. Ni, the associate editor responsible for this paper, for their many comments and suggestions which improved the quality of the paper.

Dr. Boppana's research has been partially supported by National Science Foundation Grant CCR-9208784. Dr. Chalsani's research has been supported in part by a grant from the graduate school of the University of Wisconsin-Madison, and by NSF Grants CCR-9308966 and ECS-9216308.

REFERENCES

[1] A. Agarwal et al., "The MIT alewife machine: Architecture and performance," *Proc. 22nd Ann. Int'l Symp. Computer Architecture*, June 1995.
[2] S.B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Computers*, vol. 38, pp. 555-566, Apr. 1989.
[3] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, "The Tera computer system," *Proc. 1990 Int'l Conf. on Supercomputing*, pp. 1-6, Sept. 1990.
[4] P.E. Berman, L. Gravano, and G.D. Pifarre, "Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks," *Proc. Fourth Symp. Parallel Algorithms and Architectures*, pp. 3-12, 1992.
[5] K. Bolding and L. Snyder, "Mesh and torus chaotic routing," *Proc. Advanced Research in VLSI and Parallel Systems*, 1992.
[6] R.V. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," *Proc. 20th Ann. Int'l Symp. Computer Architecture.*, pp. 351-360, May 1993.
[7] S. Borkar et al., "iWarp: An integrated solution to high-speed parallel computing," *Proc. Supercomputing '88*, pp. 330-339, 1988.
[8] A.A. Chien, "A cost and speed model for *k*-ary *e*-cube wormhole routers." Presented at Hot Interconnects 1993, Mar. 1993.
[9] A.A. Chien and J.H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *Proc. 19th Ann. Int'l Symp. Computer Architecture.*, pp. 268-277, 1992.
[10] R. Cypher and L. Gravano, "Adaptive, deadlock-free packet routing in torus networks with minimal storage," *Proc. 1992 Int'l Conf. on Parallel Processing*, pp. III-204 to III-211, 1992.
[11] W.J. Dally, "Virtual-channel flow control," IEEE *Trans. Parallel and Distributed Systems*, vol. 3, pp. 194-205, Mar. 1992.
[12] W.J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, pp. 466-475, Apr. 1993.
[13] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. 36, no. 5, pp. 547-553, 1987.
[14] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, pp.1,320-1,331, Dec. 1993.
[15] J. Duato and P. Lopez, "Performance evaluation of adaptive routing algorithms for k-ary e-cube networks," *Lecture Notes in Computer Science 853*, K. Bolding and L. Snyder, eds., pp. 45-59, Springer-Verlag, 1994.
[16] S. Felperin, L. Gravano, G. Pifarre, and J. Sanz, "Fully-adaptive routing: Packet switching performance and wormhole algorithms," *Proc. Supercomputing '91*, pp. 654-663, 1991.
[17] S.A. Felperin, L. Gravano, G.D. Pifarre, and J.L. Sanz, "Routing techniques for massively parallel communication," *Proc IEEE*, vol. 79, no. 4, pp. 488-503, 1991.
[18] E. Ganeshan and D. K. Pradhan, "Wormhole routing in de Bruijn networks," Tech. Rep., Texas A&M University, Dept. of Computer Science, College Station, Texas, Dec. 1992.
[19] C.J. Glass and L.M. Ni, "The turn model for adaptive routing," *Proc. 19th Ann. Int'l Symp. Computer Architecture*, pp. 278-287, 1992.
[20] I.S. Gopal, "Prevention of store-and-forward deadlock in computer networks," *IEEE Trans. Communications*, vol. 33, pp. 1,258-1,264, Dec. 1985.
[21] T. Gross, "Communication in iWarp systems," *Proc. Supercomputing '89*, pp. 436-445, 1989.
[22] K.D. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *IEEE Trans. Communications*, vol. 29, pp. 512-524, Apr. 1981.
[23] F. Harary, *Graph Theory*. Addison-Wesley, 1969.
[24] I.N. Herstein, *Topics in Algebra*. John-Wiley and Sons, second ed., 1975.
[25] T Hone, H. Ishihata, and M. Ikesaka, "Design and implementation of an interconnection network for the AP 1000," *Algorithms, Software Architecture*, vol. 1, pp. 555-561, Elsevier Science B.V., 1992. Information Processing 92.
[26] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.
[27] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267-286, 1979.
[28] S. Konstantinidou and L. Snyder, "The chaos router: Architecture and performance," *Proc. 18th Ann. Int'l Symp. Computer Architecture*, pp. 212-221, 1991.
[29] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Mateo, Calif.: Morgan Kaufman, 1992.
[30] S.L. Lillevik, "The Touchstone 30 Gigaflop DELTA prototype," *Sixth Distributed Memory Computing Conf.*, pp. 671-677, 1991.
[31] D.H. Linder and J.C. Harden, "An adaptive and fault tolerant wormhole routing strategy for *k*-ary *e*-cubes," *IEEE Trans. Computers*, vol. 40, no. 1, pp. 2-12, 1991.
[32] M.D. Noakes et al., "The J-machine multicomputer: An architectural evaluation," *Proc. 20th Ann. Int'l Symp. Computer Architecture*, pp. 224-235, May 1993.
[33] J. Misic, "Multicomputer interconnection network based on a star graph, *Proc. 24th Hawaii Int'l Conf. on System Sciences*, vol. 2, pp. 373-381, 1991.
[34] J.Y. Ngai and C.L. Seitz, "A framework for adaptive routing in multicomputer networks," *Proc. First Symp. Parallel Algorithms and Architectures*, pp. 1-9, 1989.
[35] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62-76, Feb. 1993.
[36] W. Oed, "The Cray research massively parallel processor system, CRAY T3D," Tech. Rep., Cray Research Inc., Nov. 1993.
[37] H. Park and D.P. Agrawal, "A novel deadlock-free routing technique for a class of de Bruijn graph based networks," *Proc. 9th Int'l Parallel Processing Symp.*, 1995.
[38] G.D. Pifarre, L. Gravano, S.A. Felperin, and J. Sanz, "Fully-adaptive minimal deadlock-free packet routing in hypercubes, meshes, and other networks: Algorithms and simulations," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, pp. 247-263, Mar. 1994.
[39] M.R. Samatham and D.K. Pradhan, "The De Bruijn multiprocessor network: A versatile parallel processing and sorting networks for VLSI," *IEEE Trans. Computers*, vol. 38, pp. 567-581, Apr. 1989.
[40] L. Schwiebert and D. N. Jayasimha, "Optimally fully adaptive routing for meshes," *Proc. Supercomputing '93*, pp. 782-791, 1993.
[41] C.L. Seitz, "Concurrent architectures," *VLSI and Parallel Computation*, R. Suaya and G. Birtwistle, eds., ch. 1, pp. 1-84. San Mateo, Calif.: Morgan-Kaufman Publishers, Inc., 1990.
[42] C.B. Stunkel et al., "Architecture and implementation of Vulcan," *Proc. 8th Int'l Parallel Processing Symp.*, pp. 268-274, Apr. 1994.
[43] H. Sullivan and T.R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine, I," *Proc. 4th Ann. Int'l Symp. Computer Architecture*, pp. 105-124, 1977.



Rajendra V. Boppana received the BTech degree in electronics and communications engineering from Mysore University, India, in 1983, the MTech degree in computer technology from the Indian Institute of Technology, Delhi, in 1985, and the PhD degree in computer engineering from University of Southern California in 1991. Since 1991 he has been a faculty member in computer science at the University of Texas at San Antonio. His research interests are in parallel computer systems, performance evaluation, computer networks, and fault-tolerant computing systems.

Suresh Chalasani received the BTech degree in electronics and communications engineering from J.N.T. University, Hyderabad, India, in 1984, the ME degree in automation from the Indian Institute of Science, Bangalore, in 1986, and the PhD degree in computer engineering from the University of Southern California in 1991. He is currently an assistant professor of electrical and computer engineering at the University of Wisconsin–Madison. His research interests include parallel architectures, parallel algorithms, and fault-tolerant systems.