In Procs. of 20th Annual Int'l Symp. on Computer Architecture, 1993

A Comparison of Adaptive Wormhole Routing Algorithms

Rajendra V. Boppana
Div. of Math. and Computer Science
The Univ. of Texas at San Antonio
San Antonio, TX 78249-0664

Suresh Chalasani
ECE Department
Univ. of Wisconsin-Madison
Madison, WI 53706-1691

Abstract. *Improvement of message latency and network utilization in torus interconnection networks by increasing adaptivity in wormhole routing algorithms is studied. A recently proposed partially adaptive algorithm and four new fully-adaptive routing algorithms are compared with the well-known* e*-cube algorithm for uniform, hotspot, and local traffic patterns. Our simulations indicate that the partially adaptive north-last algorithm, which causes unbalanced traffic in the network, performs worse than the nonadaptive* e*-cube routing algorithm for all three traffic patterns. Another result of our study is that the performance does not necessarily improve with full-adaptivity. In particular, a commonly discussed fully-adaptive routing algorithm, which uses* 2^n *virtual channels per physical channel of a* k*-ary* n*-cube, performs worse than* e*-cube for uniform and hotspot traffic patterns. The other three fully-adaptive algorithms, which give priority to messages based on distances traveled, perform much better than the* e*-cube and partially-adaptive algorithms for all three traffic patterns. One of the conclusions of this study is that adaptivity, full or partial, is not necessarily a benefit in wormhole routing.*

Keywords: *adaptive routing, deadlocks, multicomputer networks,* k*-ary* n*-cubes, message routing, store-and-forward routing, wormhole routing.*

1 Introduction

Point-to-point k-ary n-cube and related networks are being used in many recent experimental and commercial multicomputers and multiprocessors [2, 11, 26, 9, 3]. A k-ary n-cube network 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.

Routing algorithms, which specify how messages can be sent among processors, are crucial for the efficient operation of a parallel computer. For maximum system performance, a routing algorithm should have high throughput and exhibit the following important features [17]: low-latency message delivery, avoidance of *deadlocks*, *livelocks*, and starvation, and ability to work well under various traffic patterns. Since message latencies increase with increase in the number of hops, we consider only *minimal* routing algorithms as per which a message always moves closer to its destination with each hop taken; another advantage of minimal routing is that livelocks are avoided. The issue of starvation can be avoided by allocating resources such as channels and buffers in FIFO order. Ensuring deadlock-freedom is more difficult and depends heavily on the design of the routing algorithm.

Store-and-forward (SAF) [5] and *wormhole* (WH) [14] are two popular switching techniques for interconnection networks. With SAF technique, the message latency is the product of the number of hops taken and the sum of the average queuing delay and transmission time of the message per hop.

In the WH technique, a message 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. At any given time, the flits corresponding to a message occupy contiguous channels in the network. In this method, the message latency is proportional to the sum of the number of cycles spent in waiting for suitable channels to route message flits, number of hops, and message length. To avoid deadlocks, multiple virtual channels are simulated on each physical channel and a pre-defined order is enforced on the allocation of virtual channels to messages.

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. An adaptive routing algorithm can be either fully- or partially-adaptive. The well-known e-cube routing algorithm is an example of non-adaptive routing algorithms, since it has no flexibility in routing messages.

Adaptive routing algorithms have a few disadvantages, however. The complexity of the routing algorithm and, hence, the hardware cost increase with the increase in adaptivity. Furthermore, partially-adaptive routing algorithms that favor some paths more than others can cause highly uneven utilization and early saturation of the network.

Recently, several fully- and partially-adaptive algorithms for deadlock-free wormhole routing [4, 10, 15, 19, 23] have been proposed. The fully-adaptive algorithm for k-ary n-cubes by Linder and Harden [23]

uses $(n+1)2^{n-1}$ virtual channels per physical channel. The fully-adaptive wormhole algorithm by Berman *et al.* [4] for k-ary n-cubes uses as many as $10(n-1)+6$ virtual channels per physical channel. Felperin *et al.* [16] designed a fully-adaptive WH routing algorithm for tori (k-ary 2-cubes) that uses eight virtual channels per physical channel. Dally [12] proposes augmenting multicomputer networks with express channels to facilitate adaptive routing and reduce the network diameter and message latencies.

In this paper, we present results on the performance of six algorithms for uniform, hotspot, and local traffic patterns on k-ary 2-cubes. We compare a recently proposed partially-adaptive and four fully-adaptive WH routing algorithms with the commonly used e-cube algorithm. The north-last algorithm used in this paper is a member of many partially-adaptive algorithms proposed by Glass and Ni [19] based on the elegant *turn* model. One of the four fully-adaptive algorithms used in this study is based on the total number of possible directions that can be taken by a message. It is an improvement (reduces the number of virtual channels used) over the results of Linder and Harden [23] and Felperin *et al.* [16] and a generalization of the result by Dally [11] for mesh networks.

The other three fully-adaptive algorithms are derived from the store-and-forward algorithms [20] based on the the number of hops taken by messages. While routing messages, these algorithms use some form of priority information, in addition to full-adaptivity. One of these algorithms also employs load balancing of virtual channels. The design of these algorithms is based on a recent result on designing deadlock-free WH routing algorithms from SAF routing algorithms [8]. To make the paper self-contained, this method is briefly explained in the next section. For certain cases [8], these algorithms require fewer virtual channels than the previously proposed fully-adaptive algorithms.

The rest of this paper is organized as follows. Section 2 describes the routing algorithms used in our performance study. Section 3 compares the performance of six different WH routing algorithms. Section 4 concludes this paper.

2 Routing Algorithms

In this section, we discuss six different deadlock-free wormhole routing algorithms, used in this study. Out of these six algorithms four are fully-adaptive, one is partially-adaptive, and one is non-adaptive. Three of the four fully-adaptive algorithms are derived from the corresponding SAF routing algorithms, based on our recent results [8] on developing deadlock-free WH routing algorithms from SAF algorithms.

We first describe these three fully-adaptive routing algorithms, which will be collectively referred to as hop schemes.

2.1 Hop schemes

Notation. In the rest of this paper, we use k^n to denote a k-ary n-cube. Dimensions of k^n are numbered from 0 to $(n-1)$ and nodes are numbered in each dimension from 0 to $(k-1)$. Each node is uniquely indexed by an n-tuple using the n numbers it obtained in the n dimensions. We assume the adjacent nodes are connected by two unidirectional communication links. Thus, each node $x = (x_{n-1}, \ldots, x_0)$ has 2 outgoing links from it to nodes $(x_{n-1}, \ldots, x_{i+1}, x_i + 1, x_{i-1}, \ldots, x_0)$ and $(x_{n-1}, \ldots, x_{i+1}, x_i - 1, x_{i-1}, \ldots, x_0)$ in dimension i; the addition and subtraction operations performed here are with respect to modulo k. A node $x = (x_{n-1}, \ldots, x_0)$ in k^n is termed *even* (respectively, *odd*) if $\sum_{i=0}^{i=n-1} x_i$ is even (respectively, odd).

Throughout this paper, a communication channel or a communication link should be taken to mean a physical channel. Every physical channel, virtual channel, and message originating from a node can be given unique number based on the address of the node.

Construction of WH algorithms. Figure 1 illustrates construction of a WH routing algorithm from an SAF algorithm. Figure 1(a) shows the node model used for SAF routing. In SAF routing, buffers in a node are the critical resources. Deadlocks in SAF routing are avoided by partitioning the buffers into several classes — b_0, b_1, ..., b_m — and placing constraints on the set of buffer classes a message can occupy in each node. This technique of avoiding SAF deadlocks is known as the *buffer reservation* technique [20].

To derive a WH algorithm from the SAF algorithm, we proceed as follows. First, on each physical channel in the network used for WH routing, we provide virtual channels $c_0, c_1, \ldots, c_m$ and the corresponding flit-buffers (see Figure 1(b)). Next, if a message can occupy a buffer of class b_i at an intermediate node and go through a communication channel in the SAF network (see Figure 1(a)), then in the WH network the message can only reserve virtual channel c_i (see Figure 1(b)). In other words, if the SAF algorithm specifies that a message should occupy buffer of class b_i at a node and can take one channel from the set of physical channels S to complete the next hop, the corresponding WH algorithm specifies that the message at that node should take the next hop using a virtual channel of class c_i on any of the physical channels in the set S.

The above construction allows one to design a WH routing algorithm from any given SAF algorithm with

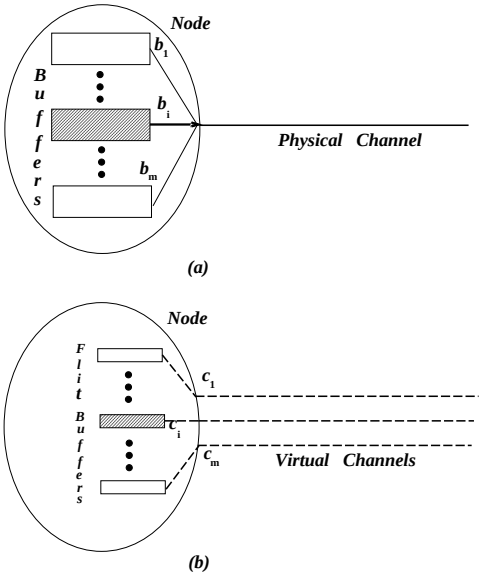

Figure 1: Derivation of WH routing from SAF routing.

the same degree of adaptivity. However, a WH algorithm derived using the above procedure need not be deadlock-free. The following lemma presents a general condition under which the WH routing algorithm designed from an SAF algorithm is deadlock-free.

Lemma 1 *If the* SAF *routing is deadlock free and the buffers occupied by every message in successive hops have monotonically increasing ranks, then the* WH *routing algorithm derived from the* SAF *algorithm is also deadlock free.*

See [8] for a proof.

A few well-known fully-adaptive SAF schemes based on the number of hops taken [20] satisfy the above lemma, and hence can be used for WH routing.

Positive-hop SAF and WH routing algorithms. In the well-known positive-hop SAF algorithm, the number of buffer classes in each node equals the diameter of the network plus one [20]. A message is placed in a buffer of class 0 in the source node. During routing, a message is placed in buffer of class i in an intermediate node if it completed i hops thus far. Since the maximum number of hops a message can take equals the diameter of the network, the maximum number of buffer classes required in each node for SAF routing equals the diameter of the network plus one; for k^n, this number equals $n\lfloor k/2 \rfloor + 1$.

The corresponding positive-hop (or, PHOP for brevity) WH routing algorithm can be designed by providing $n\lfloor k/2 \rfloor + 1$ virtual channels — $c_0, \ldots, c_{n\lfloor k/2 \rfloor}$ — on each physical link. For example, in 16^2, 17 virtual channels are provided on each physical link. A message reserves virtual channel i at an intermediate node to complete hop $i+1$. As an example, suppose that a message M with source $(4,4)$ and destination $(2,2)$ in 6^2 takes the following path:

$$(4,4) \rightarrow (3,4) \rightarrow (3,3) \rightarrow (2,3) \rightarrow (2,2)$$

This message M reserves virtual channel c_0 on the link from $(4,4)$ to $(3,4)$, c_1 from $(3,4)$ to $(3,3)$, c_2 from $(3,3)$ to $(2,3)$, and c_3 from $(2,3)$ to $(2,2)$.

The PHOP WH scheme is deadlock-free due to the following argument (see [8] for more details). In the SAF scheme, if we assign any buffer of class i a rank of i, it is easy to see that the buffer classes occupied by a message have monotonically increasing ranks. Applying Lemma 1, we conclude that the PHOP WH routing algorithm is deadlock-free.

Negative-hop (NHOP) SAF and WH routing algorithms. In the negative-hop SAF algorithm, the network is partitioned into several subsets, such that no subset contains adjacent nodes (this is the graph coloring problem). Let us assume that these subsets are labeled $1, 2, \ldots, M$ and that each node in a subset with label i is also labeled with 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 positive 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. In the negative-hop SAF scheme, a message that is currently in a buffer of class b_i can only wait for a buffer of either class b_i (if it is waiting for a positive hop) or class b_{i+1} (if it is waiting for a negative hop). Gopal [20] proves that this SAF routing is deadlock free.

For even k, the structure of k^n is a bipartite graph, and its nodes can be partitioned into two subsets (therefore, it can be colored using only two colors). Because adjacent nodes are in distinct partitions, the maximum number of negative hops a message takes is at most half the diameter of k^n, which equals $\lceil n\lfloor k/2 \rfloor/2 \rceil$. Hence, negative-hop schemes with $\lceil n\lfloor k/2 \rfloor/2 \rceil + 1$ buffer classes per node can be designed for k^n in a straightforward manner when k is even. For 16^2, for example, 9 buffer classes per node are sufficient with the negative-hop scheme. When k is odd, negative hop schemes that require about the same number of buffer classes per node can be designed [6]; however, the design of such negative-hop schemes for odd k is quite involved and will not be considered any further.

In order to derive the negative-hop WH routing algorithm from the corresponding SAF algorithm, we provide $\lceil n\lfloor k/2 \rfloor/2 \rceil + 1$ virtual channels on each physical link of k^n. When a message is generated, the total number of negative hops taken is set to zero, current

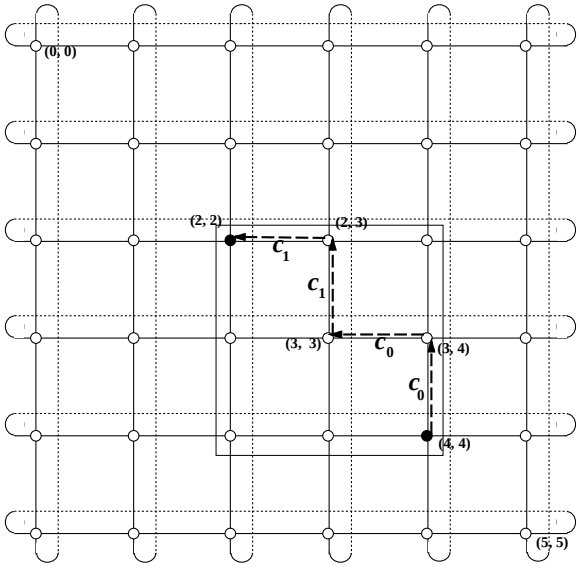

Figure 2: An example to illustrate the negative-hop scheme in 6^2.

host is set to the source node. The following pseudocode describes how a message is routed as per the negative-hop scheme.

while (*current-host* $\neq$ *destination*) **do** {

1. select the *next-host*.

2. reserve a virtual channel of class given by *number-of-negative-hops-taken* from *current-host* to *next-host*.

3. if *current-host* is odd, increment *number-of-negative-hops-taken* by one.

4. *current-host* $\leftarrow$ *next-host*.

}

A message, when it moves from an even node to an odd node, 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. Using Lemma 1, it can be shown that the negative-hop WH algorithm is also deadlock-free [8].

An example of the negative hop scheme is shown in Figure 2. In this figure, a message that originates at source node $(4,4)$ is to be routed to destination $(2,2)$ in 6^2. It can be easily shown that, as per the negative-hop scheme, the message can take any of the shortest paths from $(4,4)$ to $(2,2)$ (because of the fully-adaptive nature of the negative-hop algorithm). Suppose that the path taken by the message is

$$(4,4) \rightarrow (3,4) \rightarrow (3,3) \rightarrow (2,3) \rightarrow (2,2).$$

The hop from node $(4,4)$ to node $(3,4)$ is a positive hop and the message reserves virtual channel c_0 in order to complete this hop (recall that in all 4 virtual channels c_0, c_1, c_2, c_3 need to be multiplexed on each physical channel in order to implement the negative-hop WH scheme in 6^2). At node $(3,4)$, *current-host* is $(3,4)$ and the *number-of-negative-hops-taken* is zero, whereas the value of *next-host* becomes $(3,3)$. The message reserves virtual channel c_0 again while taking the hop from node $(3,4)$ to $(3,3)$ since the *number-of-negative-hops-taken* is still zero. However, the hop from $(3,4)$ to $(3,3)$ is a negative hop; hence, the message reserves virtual channel c_1 from node $(3,3)$ to $(2,3)$. Similarly, the message reserves virtual channel c_1 in its final hop from node $(2,3)$ to node $(2,2)$.

Negative-hop scheme with bonus cards (NBC). The negative hop (also positive hop) scheme described above does 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 numbered c_0, but only messages between diametrically opposite nodes (very few) use virtual channels numbered $c_{\lceil n\lfloor k/2\rfloor/2\rceil}$. Given below is a variation of the negative hop scheme, which attempts to achieve a more uniform utilization of virtual channels.

In the negative-hop with bonus-cards (NBC) scheme, each message is given a few *bonus cards* based on the number of negative hops it can take (which is approximately half the total number of hops it can take) before reaching destination. The number of bonus cards a message M receives at its source node in k^n is given by the following formula.

Bonus cards $=$

Maximum possible negative hops in k^n $-$

Negative hops to be taken by M.

A message with no bonus-cards is routed exactly the same as in the case of NHOP algorithm. In routing a message with b bonus cards, $b \geq 1$, any of virtual channels numbered $0, 1, \ldots, b$ can be used for the first hop of the message. Thus a message with bonus cards has a wider choice of virtual channels and is likely to choose least congested one for the first hop. The routing of message after the first hop is the same as in the case of NHOP scheme. That is, if a message arrives at an intermediate host node via a virtual channel of class c_i, then it uses a virtual channel of class c_{i+1} (if it took a negative hop to reach this node) or c_i (otherwise) to leave the node.

A more flexible version of this NBC scheme is described in [7].

2.2 Fully-adaptive routing based on the enumeration of directions

Another fully-adaptive deadlock-free WH routing scheme for a k-ary n-cube (respectively, mesh) that

uses 2^n (respectively, 2^{n-1}) virtual channels per physical channel can be derived based on the recent work of various researchers [11, 16, 23]. We refer to this algorithm as two-power-n (or, 2Pn) algorithm.

Let $s = s_{n-1} \ldots s_0$ and $d = d_{n-1} \ldots d_0$ be the source and destinations of a message being routed. Using s and d an n-bit tag $t = t_{n-1} \ldots t_0$ is created as follows.

$$t_i = \begin{cases} 1 & \text{if } s_i < d_i, \\ 0 & \text{if } s_i > d_i, \\ 0 \text{ or } 1 & \text{if } s_i = t_i. \end{cases} \quad (1)$$

Description of the algorithm. For fully-adaptive deadlock-free routing, we use 2^n virtual channels for each physical channel of the network. Each of these virtual channels is given an n-bit number. For each message to be routed, its tag is computed using (1).

In each hop, a message with tag t chooses the virtual channel with number t on any one of the links of the uncorrected dimensions[1]. □

It can be shown [8] that algorithm two-power-n routes messages free of deadlocks in k-ary n-cubes.

2.3 The North-Last Algorithm

Glass and Ni [19] proposed the north-last algorithm for multi-dimensional meshes and tori. The NLast algorithm works as follows. If destination index is less than source index in dimension 1, then a message must correct dimension 0 first before taking any hops on dimension 1 links; otherwise it is routed fully-adaptively. It prohibits adaptive routing of some messages; for example, in routing a message from node (3,3) to (1,1) in a 10^2 with upper left node being (0,0) and lower right node being (9,9), its path is always through nodes (3,2), (3,1), and (2,1) regardless of the traffic or other conditions in the network. If the four edges of a two-dimensional network are labeled West, East, North, and South, then messages that are going to North do not have adaptivity. Hence the name.

3 Simulation results

To compare the performance of these routing algorithms, we have developed an event-driven simulator. This simulator can be used for k-ary n-cubes (multi-dimensional tori) and multi-dimensional meshes for wormhole routing.

We compare the performances of six deadlock-free wormhole routing algorithms: three fully-adaptive hop schemes (positive-hop, negative-hop, and negative-hop with bonus-cards), partially-adaptive north last algorithm, proposed by Glass and Ni [19], and the well-known non-adaptive e-cube algorithm. We consider only minimal routing of messages.

To limit the search space, we have fixed some important parameters in the following performance comparisons. High-radix, $k \geq 16$, is commonly used for two- and three-dimensional networks [11, 2]. We have conducted our simulations for 16^2 tori. In literature, fixed-length messages with 16, 20, or 24 flits are commonly considered. We have considered 16-flit messages in this study. The message interarrival times are geometrically distributed.

Traffic patterns. We have considered uniform, hotspot, and local traffic patterns. The uniform (or random) traffic pattern could be representative of the traffic generated in massively parallel computations in which array data are distributed among the nodes using hashing techniques. More realistically, the traffic pattern tends to be random coupled with some local or hotspot type traffic [1, 24]. For this reason we have also performed simulations for uniform traffic coupled with a moderate hotspot traffic and completely local traffic.

In the hotspot traffic pattern simulated, a particular node receives some hotspot traffic in addition to the regular uniform traffic. For example, with a hotspot percentage of four, a newly arrived message in 16^2 is directed with 0.0438 probability to the hotspot node and with 0.0038 probability to any other node. That is, the hotspot node receives about 11.5 times more traffic than any other node in the network. In multiprocessors, this traffic pattern could be representative of computations in which critical sections (or the corresponding locks) are placed in a single node. When software techniques are used to distribute hotspot traffic, a more representative hotspot traffic is obtained by simulating multiple hotspot nodes each receiving hotspot traffic in addition to the regular uniform traffic. In this paper, we consider hotspot traffic with one hotspot node.

In the local traffic pattern, the traffic generated by node (i, j) in 16^2 is directed with equal probability to any node within the 7×7 mesh consisting of the nodes $\{(x, y) \mid i-3 \leq x \leq i+3, j-3 \leq y \leq j+3\}$. For a 16×16 torus, this corresponds to a locality factor of 0.4. This local traffic pattern is slightly different from the one considered by Agarwal [1].

Parameters of interest: latency and normalized throughput. We are interested in the average channel utilization, ρ, and average latency, l, of a message. The average latency of a message is

$$w + (m_l + \overline{d} - 1) \times f_t, \quad (2)$$

[1] The dimensions in which the message needs to take one or more hops in order to reach its destination from the current node.

where $w, m_l, \overline{d}, f_t$ are the average wait time, average length of the message in flits, average number of hops taken by a message, and the time to transfer a flit between neighbors, respectively.

For uniform traffic, the average number of hops is the average diameter of the network. For a k-ary n-cube, it is approximately $nk/4$; 16^2 has an average diameter of 8.03 for uniform traffic. The number of flits in the message is fixed at 16. It takes one clock cycle to transmit a flit between neighbor nodes. Multiple virtual channels mapped to a physical channel share its bandwidth in time-multiplexed manner; that is, $f_t = 1$.

The average channel utilization refers to the fraction of the physical channel bandwidth utilized in any time interval when the network is in steady state. It is also called the network utilization factor or normalized throughput of the network. The average channel utilization, denoted ρ, is computed as the ratio of network bandwidth utilized to the raw bandwidth available.

$$\rho = \lambda\, m_l\, \overline{d} \times \frac{\text{Number of nodes}}{\text{Number of channels}}, \qquad (3)$$

where $1/\lambda$ is the average message interarrival time. For k^n, this can be simplified to

$$\rho = \frac{\lambda\, m_l\, \overline{d}}{2n}. \qquad (4)$$

The numerator computes the average traffic generated by a node, and the denominator gives the available bandwidth due to the physical channels originating from a node.

Congestion control. If there are no restrictions placed on message injection, the network would be unusable once saturation occurs. Therefore, we have used a simple congestion control based on the one proposed for store-and-forward routing for computer networks [22]. In this method, a node is allowed to inject a message into the network if the number of messages of the same class[2] that are in the node is less than a certain specified limit. With this type of congestion control, it is feasible to simulate the network for traffic rates that would otherwise cause saturation and lead to unbounded delays. Convergence of such simulations need to be checked carefully, however.

Convergence criteria. For better randomness, separate sequences of random numbers are maintained for the distribution of message interarrival time, selection of destination, etc.

For each simulation, sufficient warmup time is provided to allow the network reach steady state. After the warmup time, the network traffic is sampled at periodic intervals. The counters used for statistics gathering are reset at the beginning of each sampling period. Statistics are gathered during the sampling time and analyzed for convergence. After each sampling period, new streams of random numbers are used for destination selection and message interarrival time, and statistics are not gathered for some period of time. Independent of the convergence criteria, a minimum of three samples and a maximum of 10-15 samples are taken for each simulation. A simulation is terminated if it exceeds the maximum time limit or if the convergence criteria are met.

For the purpose of checking convergence, we partition messages into various classes based on the number hops they require to reach destinations. One convergence check is based on the variance in the latencies reported by messages of each hop-class. This check is based on the population mean (with each hop-class being a stratum of the message population) described in [25]. For each stratum, the average latency and variance are computed. Using proper weights[3] of each stratum in the population, the average message latency, l, and variance of this average latency, ${\sigma_l}^2$, are computed. The 95% confidence interval of the average latency is given by $(l - 2\sigma_l, l + 2\sigma_l)$. The value $2\sigma_l$ is the bound on the error of estimation of l.

Another check is based on the variance of the average message latencies for each of the latest three or more samples. For this case also the bound on the error of estimation is computed. If both error bounds are within 5% of the respective averages, the simulation is terminated. For the points before saturation, these criteria are easily satisfied. Longer warmup and sampling times are needed to achieve convergence for points near and beyond saturation.

After the simulation, the average number of messages received in the sampling periods used in the above convergence criteria is computed. The bound on the error of this estimation is almost always less than 1%.

[2] For the purpose of congestion control, the class of a message is determined as follows. In the case of hop schemes and 2Pn, a message class is based on the virtual channel number it can use. In the case of e-cube and NLast schemes, a message class is based on the particular virtual channel it intends to use.

[3] The weights of each hop-class are based on the frequency with which they appear for the traffic pattern being simulated. For example, for uniform traffic on a 16^2, hop-class 1 has a weight of 0.0157 and hop-class 16 has a weight of 0.0039, since each node has four neighbors but only one diametrically opposite node. In the case of local traffic, the number of hop-classes is six: classes 1 and 6 have weight 0.0833 each, classes 2 and 5 have weight 0.1667 each, and classes 3 and 4 have weight 0.25 each.

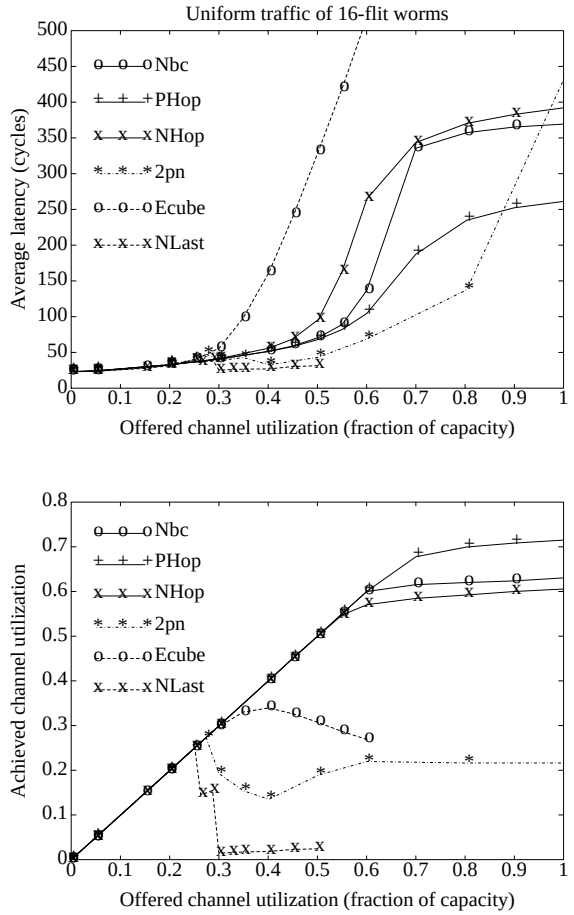

Figure 3: Performance of the routing algorithms for uniform traffic.

3.1 Simulation of uniform traffic

The average latency and normalized throughput achieved are plotted against offered traffic[4] in Figure 3 for uniform traffic with 16-flit messages. For low traffic load ($\rho \leq 0.25$), all six algorithms have the same latency. The three hop schemes and the other three algorithms behave differently during and after saturation.

The three algorithms derived from SAF routing have similar throughputs with PHop being slightly better: PHop has better latency and throughput in saturation. In particular, PHop and Nbc begin to saturate after 0.6, and NHop shows signs of saturation at about 0.55. The latencies of all three algorithms rise abruptly at the point of saturation but have bounded values, even for high traffic loads, $\rho \geq 0.75$, due to congestion control. Furthermore, the achieved throughputs of the three algorithms increase steadily. The PHop and Nbc algorithms achieve their peak throughputs of 0.72 and 0.63, respectively, at 100% offered load.

It is not meaningful to compare the saturation latencies of hop schemes with those of the other three algorithms, since the latter ones have lower throughputs. The fully-adaptive 2Pn has lower peak throughput than e-cube and saturates more quickly. The e-cube algorithm has a peak throughput of 0.34, which occurs at offered traffic of 0.4.[5] Another observation is that e-cube performs better than the partially-adaptive NLast algorithm, which is consistent with the results by Glass and Ni [19] for mesh networks. The effect of congestion control on e-cube in saturation is to limit the rate at which the latency increases and maintain throughput close to the maximum throughput slightly after the point of saturation. The congestion control seems to be less effective for 2Pn and NLast with respect to throughput but keeps message latencies low. Abrupt falls and plateau in the throughput curve of NLast indicate that the congestion control is not effective for certain traffic loads. With a different congestion control, it might be feasible to maintain its peak throughput of 0.25. (Glass and Ni [18] report a peak throughput of approximately 0.23 for this algorithm on a 10×10 mesh.)

3.2 Simulation of hotspot traffic

Performances of the six algorithms for hotspot traffic with 4% hotspot traffic are given in Figure 4. The hotspot node is chosen to be node $(15, 15)$. We have experimented with various different choices for hotspot nodes and found that the NLast yields best results when the hotspot node is $(15, 15)$; performances of the e-cube and hop schemes are unaffected by the choice of the hotspot node.

Compared to uniform traffic, the increase in latencies due to hotspot traffic is negligible when the traffic is low (applied load of 0.2 or less). However, hotspot

[4] Normalized throughputs are considered in performance comparisons.

[5] To verify the validity of our simulations for e-cube, we compared the peak throughput reported here with those indicated in various previous studies. Song [27] reports peak channel utilizations of 40% for e-cube routing on mesh networks with bidirectional (half-duplex) channels. He also shows that the use of two unidirectional channels to connect adjacent nodes results in lower throughputs. Since we simulate two unidirectional channels for connections between nodes, our results are correspondingly lower. Though torus is slightly different from a two-dimensional mesh, our experiments indicate that e-cube yields similar normalized throughputs on these networks when other parameters such as network size and message length are kept the same. Berman *et al.* [4] simulate a variant of e-cube (which allows non-minimal routing) and report a peak throughput of 0.2 for message population consisting of a mix of 15- and 31-flit messages on a 31×31 torus. The often reported 50% network utilization obtained by e-cube on mesh networks [26, 13] is based on the bisection bandwidth width arguement, which actually gives a better upper bound on the available bandwidth for meshes with uniform traffic. This value cannot be used to compare channel utilizations, however.

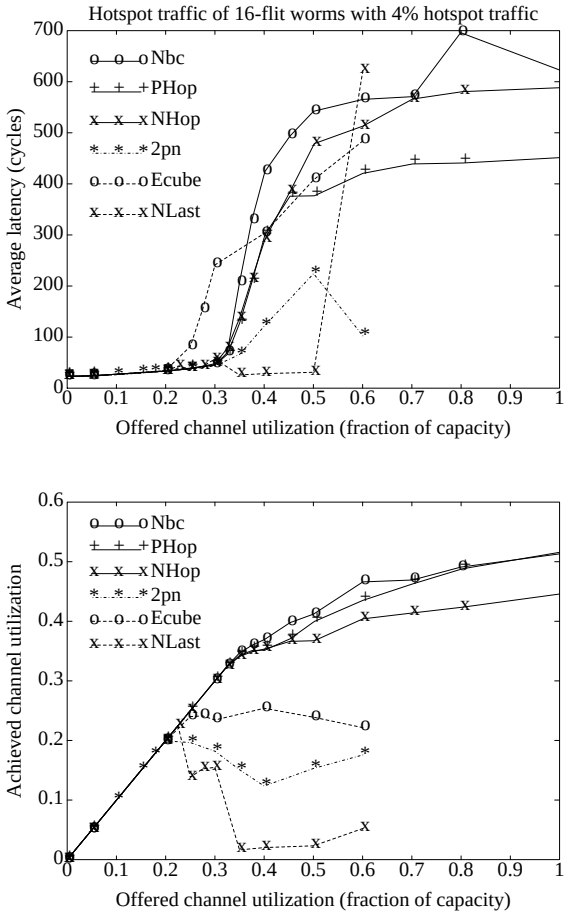

Figure 4: Performance of the routing algorithms for 4% hotspot traffic.

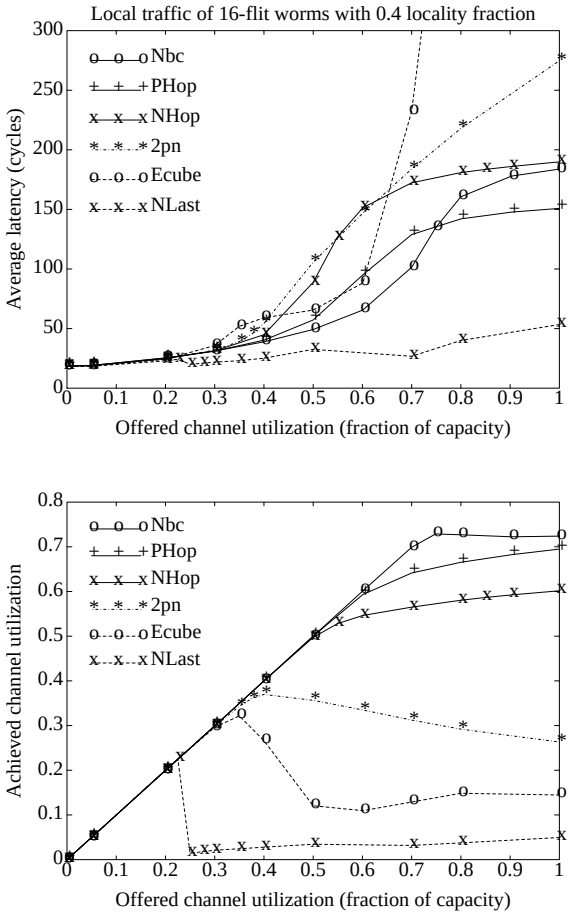

Figure 5: Performance of the routing algorithms for local traffic with 0.4 locality factor.

traffic causes early saturation and latencies in saturation are much higher compared to the uniform traffic case.

Once again, *e*-cube, NLast, and 2Pn algorithms saturate much earlier than the hop schemes. Of these three, *e*-cube is the best algorithm yielding a maximum normalized throughput of 0.25. It is consistently better than 2Pn and NLast algorithms. The peak normalized throughput realized by PHop and Nbc is slightly more than 0.5, while that of NHop is about 0.45. The actual saturation for these algorithms begins at about 0.35. But due to congestion control, the latencies are controlled and normalized throughputs increase steadily.

3.3 Simulation of local traffic

Figure 5 presents the performances of the six algorithms for local traffic with 0.4 locality of fraction. In this case, 2Pn with a peak throughout of 0.37 performs better than *e*-cube, and NLast has the least throughput. Hop schemes have much higher normalized throughput and controlled latencies in saturation region. A noteworthy point is that Nbc with peak throughput of 0.72 performs better than PHop, since it uses more virtual channels (due to bonus cards) in routing messages. It also has the lowest latency among the three hop schemes for up to 0.75 applied traffic load.

3.4 Discussion

The partially-adaptive NLast algorithm does not exhibit better performance than the simpler *e*-cube algorithm. It causes early saturation of the network for the three traffic patterns. It requires complicated routing logic, which could increase the node complexity, node delay per hop, or both. The main problem with the NLast algorithm is that it skews even uniform traffic. However, Glass and Ni [19] report that this class of algorithms perform better than *e*-cube for other types of nonuniform traffic such as matrix transpose.

Our results indicate that the fully-adaptive hop schemes PHop, Nbc, and NHop yield better throughputs for the traffic patterns considered in this study. This could be due to the use of more virtual chan-

nels per physical channel [13], balancing the traffic on virtual channels, or both. For example, for uniform traffic, PHop gives better throughput and also uses more virtual channels than any other algorithm. In the case of hotspot traffic, however, Nbc gives better throughput than PHop despite the use of fewer virtual channels. A possible explanation is that the load on virtual channels is balanced in Nbc but not in NHop and PHop algorithms.

The fully-adaptive 2Pn scheme performs worse than the non-adaptive *e*-cube algorithm for uniform and hotspot traffic. Trying to explain the unexpected low performance of the 2Pn algorithm, we have closely looked at algorithms Nbc and 2Pn. In this light, we have simulated 2Pn and Nbc algorithms for virtual-cut-through routing [21] of 16-flit packets on 16^2 for uniform traffic. The 2Pn algorithm performed as well as Nbc and better than *e*-cube with respect to both latency and peak throughput.

The 2Pn scheme routes a message at each hop based on the local knowledge available at the current host node. In contrast, Nbc uses local knowledge with some kind of priority information (based on the number of hops taken) to route a message. In the case of packet routing, lack of this information is not a severe handicap to 2Pn, since it employs one-hop lookahead (which is known to yield close to optimal performance for uniform traffic), and the penalty for not choosing the best path is not too severe. In the case of wormhole routing, however, the critical resources (channels) are held for a longer time. Therefore, the penalty for choosing a path that later turns out to be congested is more severe for wormhole routing than for SAF routing. This indicates that the use of priority could be beneficial in wormhole routing.

4 Concluding remarks

In this paper, we have evaluated the effectiveness of several WH routing algorithms with various degrees of adaptivity. We have considered four fully-adaptive (PHop, NHop, Nbc, and 2Pn) algorithms, one partially-adaptive (NLast) algorithm, and the well-known non-adaptive *e*-cube algorithm. Of the four fully-adaptive algorithms, 2Pn uses the fewest virtual channels, four, for tori.

The remaining three fully-adaptive algorithms, PHop, NHop, and Nbc, are obtained from hop based packet routing algorithms. Algorithm PHop uses as many as 17 virtual channels per physical channel for 16×16 tori. Algorithms NHop and Nbc use nine virtual channels per physical channel. These hop schemes are different from the other three algorithms, since they use some sort of priority information in routing messages. Furthermore, algorithm Nbc tries to balance the load on virtual channels, a feature not given much attention previously. The purpose of our study is to see the improvements in network throughput and message latency with the use of adaptivity and other features. Our observations are summarized as follows.

Fully-adaptive algorithms do not necessarily yield better throughput than non-adaptive algorithms. A case in point is the 2Pn fully-adaptive algorithm. For two-dimensional tori, it looks very attractive, since it provides full-adaptivity with only four virtual channels per physical channel. However, simulation results show that *e*-cube algorithm outperforms it for uniform and hotspot traffic patterns. On the other hand, the other three fully-adaptive algorithms based on hop schemes, namely, PHop, NHop, and Nbc algorithms, require more virtual channels but improve throughputs substantially. For the traffic patterns used, hop schemes are better than the other three algorithms. A comparison of NHop and Nbc algorithms indicates that balancing traffic on virtual channels yields higher throughput and lower message latency for a given throughput.

Another point of observation is that partially-adaptive algorithms such as NLast [17, 19] do not compare well with *e*-cube and hop-based algorithms. They have routing logic complexity comparable to that of a fully-adaptive algorithm and performance similar to or worse than that of the non-adaptive *e*-cube algorithm. This observation does not apply to the planar adaptive routing proposed by Chien and Kim [10], since it is an entirely different type of partially-adaptive algorithm.

We are conducting further simulations of these routing algorithms for multidimensional tori and meshes. In future, we intend to use communication traces obtained from computations on parallel processors to evaluate the performances of routing algorithms. Another issue of interest is evaluation of improvements in throughputs with addition of virtual channels. Dally [13] shows that additional virtual channels improve the performance of *e*-cube for uniform traffic. Our work here indicates that the use of priority is beneficial in fully-adaptive routing. Further work is needed, for example, to see if the extensive amount of priority information used by PHop is indeed necessary. The issue of balancing load on virtual channels needs to be explored further. Though the number of virtual channels used in hop schemes is a concern, there are ways to reduce this number substantially for NHop based algorithms [6]. We are also studying the implementation aspects of these algorithms.

Acknowledgements

The authors thank Profs. C.S. Raghavendra and D.K. Panda for many discussions and comments on an ear-

lier draft of this paper, Prof. Ram C. Tripathi for discussions on the convergence criteria used in the simulations, and Mr. Francis Ho for help in developing the simulator. The first author's research is supported by NSF Grant CCR-9208784, and the second author's research by a grant from the Graduate School of UW-Madison. Most of the simulations have been performed on the workstations in the CS Lab at UT-San Antonio supported by NSF Grant USE-950407 under ILI program.

References

[1] A. Agarwal. Limits on interconnection network performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):398–412, Oct. 1991.

[2] A. Agarwal, *et. al.* The MIT Alewife machine: A large-scale distributed multiprocessor. In *Proc. of Workshop on Scalable Shared Memory Multiprocessors*. Kluwer Academic Publishers, 1991.

[3] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The Tera computer system. In *Proc. 1990 Int. Conf. on Supercomputing*.

[4] P. E. Berman, L. Gravano, and G. D. Pifarre. Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks. In *Proc. Fourth Symposium on Parallel Algorithms and Architectures*, pages 3–12, 1992.

[5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall Inc., 1987.

[6] R. V. Boppana and S. Chalasani. Design of hop-based wormhole routing algorithms with reduced virtual channel requirements. In preparation.

[7] R. V. Boppana and S. Chalasani. A comparison of wormhole routing algorithms based on adaptivity. Technical Report UTSA-CS-92-113, University of Texas at San Antonio, Division of Math., Comp. Sci., and Statistics, San Antonio, Texas, Nov. 1992.

[8] R. V. Boppana and S. Chalasani. New wormhole routing algorithms for multicomputers. Technical report, Univ. of Wisconsin-Madison, Dept. of Electrical and Computer Engineering, Madison, WI, 1992. Some of the results will be presented at the 7th Int. Parallel Processing Symposium, 1993.

[9] S. Borkar *et al.* iWarp: An integrated solution to high-speed parallel computing. In *Proc. Supercomputing '88*, pages 330–339.

[10] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proc. 19th Ann. Int. Symp. on Comput. Arch.*, pages 268–277, 1992.

[11] W. J. Dally. Network and processor architecture for message-driven computers. In R. Suaya and G. Birtwislte, editors, *VLSI and Parallel Computation*, chapter 3, pages 140–222. Morgan-Kaufman Publishers, Inc., San Mateo, California, 1990.

[12] W. J. Dally. Express cubes: Improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 40(9):1016–1023, Sept. 1991.

[13] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, Mar. 1992.

[14] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, C-36(5):547–553, 1987.

[15] J. Duato. On the design of deadlock-free adaptive routing algorithms for multicomputers: Theoretical aspects. In *PARLE '91: Parallel Architectures and Languages*, pages 234–243.

[16] S. Felperin, L. Gravano, G. Pifarre, and J. Sanz. Fully-adaptive routing: Packet switching performance and wormhole algorithms. In *Proc. Supercomputing '91*, pages 654–663.

[17] S. A. Felperin, L. Gravano, G. D. Pifarré, and J. L. Sanz. Routing techniques for massively parallel communication. *Proceedings of the IEEE*, 79(4):488–503, 1991.

[18] C. J. Glass and L. M. Ni. Adaptive routing in mesh-connected networks. In *Proc. Int. Confernce on Distributed Computing Systems*, pages 12–19, 1992.

[19] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *Proc. 19th Ann. Int. Symp. on Comput. Arch.*, pages 278–287, 1992.

[20] I. S. Gopal. Prevention of store-and-forward deadlock in computer networks. *IEEE Trans. on Communications*, COM-33(12):1258–1264, Dec. 1985.

[21] P. Kermani and L. Kleinrock. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks*, 3:267–286, 1979.

[22] S. S. Lam and M. Reiser. Congestion control of store-and-forward networks by input buffer limits—an analysis. *IEEE Trans. on Communications*, com-27(1):127–133, Jan. 1979.

[23] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. on Computers*, 40(1):2–12, 1991.

[24] G. F. Pfister and V. A. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Trans. on Computers*, c-34(10):943–948, Oct. 1985.

[25] R.L. Scheaffer, et. al. *Elementary Survey Sampling*. Duxbury Press, North Scituate, Mass., 2 edition, 1979.

[26] C. Seitz. Concurrent architectures. In R. Suaya and G. Birtwislte, editors, *VLSI and Parallel Computation*, chapter 1, pages 1–84. Morgan-Kaufman Publishers, Inc., San Mateo, California, 1990.

[27] P. Y. Song. Design of a network for concurrent message passing systems. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1988.