In Procs. 7th Int'l Parallel Processing Symposium 1993

New Wormhole Routing Algorithms for Multicomputers

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. *Development of wormhole routing techniques so far has been largely independent of the results available for store-and-forward routing in literature. In this paper, we provide a general result which enables us to design deadlock-free wormhole routing algorithms from store-and-forward routing algorithms that satisfy certain criteria. We illustrate this result by developing fully-adaptive deadlock-free wormhole routing algorithms from two well-known store-and-forward algorithms: the positive- and negative-hop algorithms based on the number of hops taken by messages. We compare the negative-hop algorithm with the commonly used non-adaptive* e-*cube and recently proposed partially adaptive north-last algorithm.*

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 networks are currently being used in many recent experimental and commercial multicomputers. A k-ary n-cube multicomputer 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 Ncube Series 2 (hypercube) systems and Intel Paragon (a mesh computer system) are two commercially available k-ary n-cube systems.

Routing algorithms, which specify how messages can be sent among processors, are crucial for the efficient operation of a multicomputer. For maximum system performance, a routing algorithm should exhibit the following important features [9]: low-latency message delivery, avoidance of *deadlocks* and *livelocks*, and ability to work well under varying 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. Deadlocks correspond to states in which a set of messages are forever blocked, whereas livelocks represent states in which one or more messages could be forever denied of the resources they require to progress towards their destinations. Livelock-freedom can, in general, be ensured by assigning resources (channels or buffers) to waiting messages in a FIFO manner. Ensuring deadlock-freedom is more difficult and depends heavily on the design of the routing algorithm.

Store-and-forward (SAF) [2] and *wormhole* (WH) [6] 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 flits. A communication channel in the network, once it transmits the first flit of a message, must also transmit all the remaining flits of the same message before it can accept flits of another message. Thus, 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 while routing the first flit (header), the number of hops, and the message length. To avoid deadlocks in wormhole routing, multiple virtual channels are simulated on each physical channel and a predefined order is enforced on the allocation of virtual channels to messages. A routing algorithm which, in general, allows a source-destination pair to use more than one path when routing messages is called an *adaptive* routing algorithm. A routing algorithm is termed a *fully-adaptive minimal* routing algorithm if any possible minimal path between a source and destination can be potentially used at the time messages are injected into the network.

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, which favor some paths in the network more than others, can cause highly uneven utilization and early saturation of the network.

Recently, several fully- and partially-adaptive algorithms for wormhole routing [1, 5, 7, 10, 12] have been proposed. Linder and Harden proposed a deadlock-free fully-adaptive wormhole routing algorithm for k-ary n-cubes that uses $(n+1)2^{n-1}$ virtual channels per physical channel [12]. Berman *et al.* devised a fully-adaptive WH algorithm for k-ary n-cube that, in general, uses $10(n-1)+6$ virtual channels per physical channel [1]. Felperin *et al.* designed a fully-adaptive

WH routing algorithm for tori (k-ary 2-cubes) that uses four virtual channels per physical channel [8].

In this paper, we derive a general result to show that a class of SAF routing algorithms can be used, with appropriate modifications, for WH routing. As an example of this result, we derive a deadlock-free WH algorithm from an SAF algorithm based on the number of hops taken by messages. For k-ary n-cube networks with radix, k, 40 or less, this new WH routing algorithm requires fewer virtual channels than the recently proposed fully adaptive WH scheme in [1]. Also, the proposed routing algorithm requires fewer virtual channels than the fully-adaptive WH method of Linder and Harden [12], when k is less than $\frac{n+1}{n}2^{n+1}$ (for example, when $k < 20$ for 3-dimensional tori).

The rest of this paper is organized as follows. Section 2 presents the result on developing WH routing algorithms from SAF algorithms. This section also presents two new WH routing algorithms based on this result and compares them with the e-cube algorithm and a recently proposed WH routing algorithm. Section 3 concludes this paper.

2 Application of SAF algorithms for WH routing

In this section, we establish the correspondence between SAF and WH routing algorithms — conditions under which one can derive a corresponding (deadlock-free) WH routing algorithm for an SAF routing algorithm. The SAF routing algorithms considered in this paper avoid deadlocks using *buffer reservations*. In a buffer reservation method, the buffer pool in a node is partitioned into several classes of buffers and, in order to avoid deadlocks, each message is permitted to occupy only a restricted set of buffer classes. Though there exist a plethora of deadlock-free SAF routing algorithms based on buffer reservation in the computer networks area, we consider only a few classes of those algorithms in this paper [11].

We indicate a k-ary n-cube as k^n and assume that neighbor nodes are connected by two physical channels, one for each direction of communication.

Construction of WH algorithms. Given a SAF routing algorithm, we derive a corresponding WH routing algorithm as follows. If a message can occupy a buffer of class $b_0, b_1, \ldots$, or b_m at an intermediate node and go through a communication channel (or, equivalently, physical channel) then virtual channels named $c_0, c_1, \ldots, c_m$ are provided for WH routing on that communication channel. If a message occupies a buffer of class b_i at the intermediate node j and takes communication channel between nodes j and k in SAF routing, then, in the corresponding WH routing, the header flit of the message acquires virtual channel c_i in the communication channel connecting j and k. 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 following lemma presents a condition under which a WH routing algorithm derived from an SAF algorithm is deadlock free (see [4] for a proof). Buffers in a network can be ranked such that there exists a partial or full order among them.

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.*

Our method facilitates the development of an equivalent WH routing algorithm from any SAF routing algorithm. However, not all SAF algorithms yield deadlock-free WH routing algorithms. We show below that a couple of well-known SAF schemes based on the number of hops taken [11] satisfy the above lemma, and hence can be used for WH routing. In the following discussion, we assume that a message never waits for a buffer in its destination and that it is consumed immediately upon arrival at its destination.

The positive-hop WH algorithm. In the positive-hop SAF algorithm, the number of buffer classes in each node is one plus the maximum number of hops taken by a message. Thus, for minimal routing on k-ary n-cubes the number of buffer classes equals $1 + n\lfloor k/2 \rfloor$. Let these buffer classes be labeled $b_0, \ldots, b_{n\lfloor k/2 \rfloor}$. A message can occupy a buffer of class b_i if it has completed i hops. A message holding a buffer of class b_i can only wait for a buffer of class b_{i+1}. This algorithm is deadlock free with SAF switching [11].[1] To show that the WH algorithm derived from it is also deadlock free, we need to show that there exists a ranking of buffers such that Lemma 1 is satisfied. All buffers of class b_i has rank i. Therefore, any buffer of class i has a higher rank than any buffer of class $i-1$ or lower, but there is no order defined among buffers of a class. Then, there is a partial

[1] There are other variations of this positive-hop SAF algorithm. One such variation is to allow a message to occupy any buffer in classes $0, \ldots, i$, if it completed i hops. This variation of the algorithm is deadlock free but does not satisfy Lemma 1. Therefore, it cannot be used for deadlock-free WH routing.

ordering of buffers in the network and the sequence of buffers a message occupies during its journey have monotonically increasing ranks. Applying Lemma 1, the positive-hop WH algorithm derived from the above SAF algorithm is also deadlock free.

The corresponding positive-hop WH algorithm is constructed as follows. Corresponding to each physical channel in the network, there are $n\lfloor k/2 \rfloor + 1$ virtual channels: $c_0, \ldots, c_{n\lfloor k/2 \rfloor}$. At any intermediate node, a message can reserve virtual channel c_i if and only if it has taken i hops to reach the intermediate node. As an example, let us consider a 16^2 (16×16 torus). To implement the positive-hop SAF algorithm, each node should have 17 different classes of buffers. Therefore, in the positive-hop WH algorithm, 17 virtual channels need to be multiplexed on each physical channel.

The negative-hop WH algorithm. In the negative-hop SAF algorithm [11], the network is partitioned into several subsets, such that no subset contains two adjacent nodes (this is the graph coloring problem). Further, 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 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 needs to take a positive hop) or class b_{i+1} (if it needs to take a negative hop) to reach the next node. Gopal [11] proves that this SAF scheme 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 when k is even. A similar result holds for the case k is odd; however, the design of such negative-hop schemes for odd k is quite involved and will not be discussed here. In the rest of this paper, negative-hop schemes are discussed for k^n with even k.

An advantage of the negative-hop scheme is that it requires fewer buffer classes than the positive-hop scheme. For 16^2, for example, the positive-hop scheme requires 17 buffer classes per node, where as the negative-hop scheme needs only 9 buffer classes.

In order to derive the negative-hop WH routing algorithm from the corresponding SAF algorithm, we need to rank buffer classes and show that the SAF version satisfies Lemma 1. For negative-hop scheme, ranking buffer classes is not as straightforward as in the case of the positive-hop scheme owing to the following reason: a message in this algorithm moves from one buffer class to another only if it takes a negative hop and continues to remain in the same buffer class as long as it takes positive hops only. Thus we need to rank not only different buffer classes, but also buffers of the same class in different nodes. We rank buffer classes such that buffer class b_{i+1} receives a rank higher than buffer class b_i regardless of the nodes in which they reside; furthermore, buffer class b_i in node x is given a higher rank than buffer class b_i in node y if node x has a higher label than node y. In essence, the rank of a buffer can be considered as a two-tuple: (b, l), where b is the class of the buffer and l is the partition label or color of its node. All buffers of a class in a node are considered equivalent and there is no order defined among them. Given two buffers, (b_i, l_i) and (b_j, l_j), then

$$\text{Rank of } (b_i, l_i) > \text{Rank of } (b_j, l_j), \quad \text{if } b_i > b_j \text{ or } b_i = b_j \text{ and } l_i > l_j.$$

If $b_i = b_j$ and $l_i = l_j$, then the rank between (b_i, l_i) and (b_j, l_j) is not defined.

Due to this ranking and the fact that successive hops of a message alternate between positive and negative hops, the buffers occupied by any message in successive hops in the SAF routing algorithm have monotonically increasing ranks. Applying Lemma 1, the corresponding negative-hop WH routing algorithm (denoted, NHop) is deadlock free.

2.1 Simulation Results

In this section, we compare performances of the negative-hop (NHop), e-cube, and the partially-adaptive *north-last* (NLast) [10] WH routing algorithms.

Parameters of interest. We are interested in the average channel utilization, ρ, average latency, l, and average wait time, w, of a message. The average time taken for transmission of a message is $w + (m_l + \overline{d} - 1) \times f_t$, 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 $nk/4$; for 16^2 it is 8.03. The number of flits in the message is fixed. It takes one clock cycle to transmit a flit between neighbor nodes using full physical channel bandwidth. However, if the physical channel

bandwidth is multiplexed among, say, 4 virtual channels, then it takes four cycles to transmit a flit between neighbors, and $f_t = 4$.[2] Message interarrival times are geometrically distributed λ as the average message arrival rate. The average channel utilization, ρ, refers to fraction of the link bandwidth utilized in any time interval. It is also the normalized throughput of the network, the ratio of network bandwidth utilized to the raw bandwidth available.

$$\rho = \frac{\lambda \, m_l \, \overline{d}}{2n} \qquad (1)$$

The numerator computes the average traffic generated by a node, and the denominator gives the available bandwidth due to the links originating from the node. This definition of channel utilization is common in computer networks literature. In parallel processing community, normalized utilization in terms of bisection bandwidth is commonly used. While both definitions are equivalent for node and edge symmetric networks such as tori, the latter does not represent the channel utilization for networks such as meshes.

Traffic patterns and message length. We have performed simulations for 16^2 tori with uniform and hotspot traffic. Due to space constraints, we report simulation results only for the case of 4-flit messages.

The simulator and simulations. To compare performance of these routing algorithms, we have developed an event-driven simulator. Startup effects are eliminated by resetting the counters used for gathering statistics after the network has reached steady state. For each point in the graphs reported here, we have used one million simulation cycles for the network to reach a steady state. Then, the simulation is run for another million cycles in the steady state and statistics are gathered during these latter one million simulation-cycles; during this period, for each source-destination pair, a minimum of six thousand (for high traffic rates, as many as 25 thousand) messages are generated and delivered.

Results for uniform traffic. The average latency in cycles is plotted against normalized throughput (or, channel utilization) for uniform traffic in Figure 1 for four-flit messages. For low traffic load, $\rho \leq 0.12$, the e-cube and the NLast algorithms are almost identical, with NLast being slightly better. However, the NLast algorithm, which is partially-adaptive, leads to network saturation at $\rho = 0.13$, whereas the non-adaptive e-cube remains competitive for $\rho \leq 0.17$. This observation is consistent with the results reported by Glass and Ni [10], who explain that NLast skews the traffic and creates non-uniformness. Compared to e-cube, NLast improves message latency by less than 4% at low traffic and saturates the network at 25% lower channel utilization. The NHop algorithm exhibits a higher average latency than both e-cube and NLast algorithms for the following reasons: each virtual channel is allocated a fixed bandwidth, in our simulations, and the NHop algorithm uses 9 virtual channels per physical link whereas the NLast and e-cube algorithms use two virtual channels per physical link. Therefore, the average latency for the NHop scheme is roughly 4.5 times that given by the the e-cube (or, NLast) algorithm for low traffic loads. Due to its fully-adaptive nature, however, the NHop algorithm performs better than both e-cube and NLast algorithms for moderate traffic (when ρ is about 0.2). For example, with 4-flit worms, the channel utilization at saturation of the NHop scheme is 0.255 which is roughly 100% (respectively, 46%) more than that for the NLast (respectively, e-cube) algorithm. We have obtained similar results [4] for different worm lengths (16-flit, 8-flit, etc.) for various tori, 16^2 and 10^2.

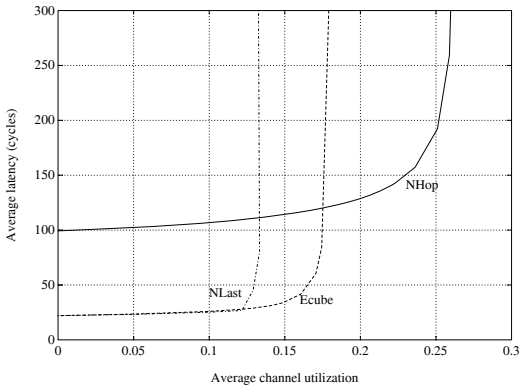

Figure 1: Average channel utilization vs. average message latency for wormhole routing of four-flit size messages on a 16×16 torus for uniform traffic.

Results for hotspot traffic. We have also simulated the performance of the three algorithms for hotspot traffic. In Figure 2, simulation results are reported for 16^2 for all three algorithms. For the results reported in this figure, we have used message lengths of 4 flits and a hotspot traffic of 4 percent. With a hotspot percentage of four, a newly arrived message in 16^2 is directed with 0.0438 probability to the hotspot node $(15, 15)$ and with 0.0038 probability to every other node. (That is, the hotspot node receives about 11.5 times more traffic than other nodes in the network.) The hotspot node is chosen to be node $(15, 15)$, which is the node at the lower right

[2] The simulations reported here are done for the case where each virtual channel has a fixed bandwidth allocated to it.

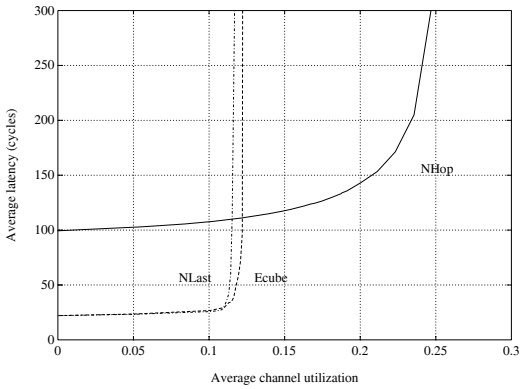

Figure 2: Average channel utilization vs. average latency for wormhole routing of four-flit size messages on a 16×16 torus for 4% hotspot traffic.

corner of 16^2. We have experimented with various different choices for hotspot nodes and have found that the NLast (which is partially-adaptive) yields best results when the hotspot node is $(15, 15)$; performance of NHop (which is fully-adaptive) and e-cube (which is non-adaptive) algorithms is independent of the choice of the hotspot node.

The simulation results show that the NLast and e-cube algorithms saturate when channel utilization reaches 11.4% and 12.2% respectively. The NHop algorithm, however, reaches saturation at $\rho = 23.5\%$, which is roughly 106% (respectively, 93%) more than the saturation channel utilization yielded by the NLast (respectively, e-cube) algorithm. The channel utilization at saturation for each algorithm is lower for hotspot traffic than that for uniform traffic when all other parameters are kept the same.

3 Concluding Remarks

In this paper, we have provided a general framework for converting deadlock-free SAF routing algorithms into deadlock-free WH routing algorithms. Using this framework, we have designed two new fully-adaptive WH routing algorithms — known as the positive-hop and the negative-hop algorithms — for k-ary n-cubes. In addition, we have simulated the NHop algorithm and compared its performance against the non-adaptive e-cube and partially-adaptive NLast algorithms for uniform and hotspot traffic patterns. Our simulation results indicate that the NHop algorithm offers higher channel utilization at saturation and offers better delays for moderate traffic. Though we have used fixed bandwidth allocation for virtual channels, the relative performances of the algorithms are likely to be the same even when bandwidth is allocated to virtual channels on demand (this observation holds for virtual-cut-through routing using these algorithms). Further results on performances of other fully- and partially adaptive algorithms for various types of traffic patterns are given in [3].

Acknowledgements

The authors thank Prof. C.S. Raghavendra for many preliminary discussions on this work and pointing us to a key paper, by Gopal [11], used in this work. 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.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.