

The Red Rover Algorithm for Deadlock-Free Routing on Bidirectional Rings

Jeff Draper

USC/Information Sciences Institute

4676 Admiralty Way

Marina del Rey, CA 90292

(310)822-1511 x750

Email: draper@isi.edu, FAX: (310)823-6714

Abstract

An algorithm for deadlock-free routing in bidirectional ring structures in multicomputer networks is presented. This algorithm provides greater throughput and lower message latencies than those of previously presented ring routing algorithms. Insight into these results is given by a message traffic analysis, and increased performance for wormhole-routed networks is quantified through simulation experiments. Additionally, a routing element which implements this algorithm is shown to be simpler and faster than that for a currently-used algorithm.

Keywords: Multicomputer Networks, Routing, Virtual Channels

1 Introduction

First generation distributed-memory multicomputers such as the Cosmic Cube [10] used “store-and-forward” packet-switching methods for routing messages among the multicomputer nodes. The present generation of multicomputers is able to reduce hardware communication overheads of first-generation machines by over two orders of magnitude by employing innovative routing techniques based on *virtual cut-through routing* [8] or its blocking variant, *wormhole routing* [1]. Wormhole routing is a pipelined, circuit-switching mechanism that is being used in several multicomputers such as the Intel Paragon [6] and the Cray T3D [2]. It is particularly popular because it has a simple hardware implementation that can provide for high-bandwidth/low-latency communication. Unfortunately, wormhole routing has the blocking property characteristic of circuit-switched networks. Consequently, its performance degrades rapidly as the message traffic increases beyond some level as a high percentage of the messages get blocked before they can reach their destination. The other drawback of a blocking technique is

the possibility of deadlock if the routing scheme is not appropriately constrained [3]. A good introduction to various aspects of wormhole routing is found in [9].

Another trend in multicomputers is the use of the ring as the interconnection network, either as the entire routing network or as a subgraph of the network. Perhaps the first mention of the ring structure for use in a multicomputer was the design of the Torus Routing Chip (TRC) [4]. More recently, the KSR1 multicomputer employed a hierarchy of rings to serve as a communications backbone [7]. Also, the Cray T3D contains a 3D torus, where any contiguous set of nodes along a dimension are connected by a ring [2].

The ring and torus connections are starting to gain favor over mesh-connected (no wraparounds) networks primarily because the diameter of a ring is half that of a linear strip, which contains no wraparound connections. As machines scale up to large numbers of nodes, the decrease in latency that a torus provides over a mesh becomes significant. Given the recent popularity of ring structures, it is important that the ring be used efficiently. One commonly used routing algorithm for multicomputer rings is one which uses virtual channels to embed a spiral in the ring to achieve deadlock-free routing. This scheme was introduced with the design of the TRC [4] and is used on the Cray T3D [2]. While this algorithm is appropriate for unidirectional rings, other allocations are possible for bidirectional rings.

In this paper, the Red Rover algorithm for deadlock-free routing on bidirectional rings is presented. The algorithm is applied to wormhole-routed networks in this paper, although it can also be used with store-and-forward routing. Like the spiral algorithm, this scheme uses virtual channels to effect deadlock-free routing. However, it prescribes a different allocation of the virtual channels—one which provides higher throughput/lower latency and results in a simpler hardware implementation. Since a ring is the basis for torus-connected k -ary n -cubes, i.e., a ring is simply a k -ary 1-cube, the algorithm can easily be applied to general k -ary n -cubes. First, background information is given in Section 2. Section 3 presents the Red Rover scheme and contrasts it to the spiral routing algorithm. A performance evaluation based on message traffic analysis is discussed in Section 4 and validated by simulation experiments reported in Section 5. Lastly, some comments are given in Section 6, followed by concluding remarks in Section 7.

2 Background

Each multicomputer node consists of a processing element (PE) with local memory and a routing element (RE), as shown in Figure 1. The system considered in this paper is a k -node bidirectional ring. The unidirectional channels that comprise the bidirectional ring are labeled, as indicated in Figure 1, with a number and an operator. The number i corresponds to the node number from which the channel emanates and the operator (+ or -) indicates the direction of the channel, e.g., a channel labeled with a + carries data from node i to node $(i + 1)$. The range of i is from 0 to $k - 1$, so all operations on i are mod k . The RE physically has 3 output channels and 3 input channels, and connections between inputs and outputs are made through a switching fabric contained within the RE. Each node is connected to each of its two neighbors by one input channel and one output channel. The remaining input and output channel

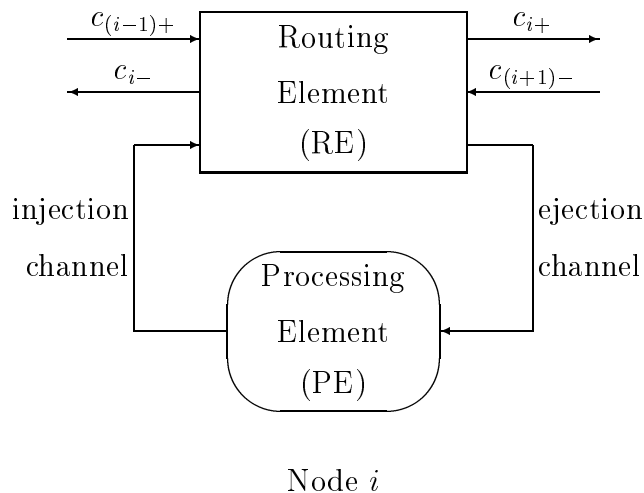


Figure 1: Node organization

of the RE are used by the node to inject/receive messages into/from the network. The switching network of the RE can simultaneously connect multiple inputs to multiple outputs, given there is no contention among output requests. The RE contains one flit buffer for each incoming network virtual channel, where a virtual channel is merely a logical channel and multiple virtual channels may share the same physical channel. The PE is assumed to have infinite buffer space for queuing up messages for the injection channel and receiving messages from the ejection channel.

Flow control within the network is based on wormhole routing. In wormhole routing, each message is segmented into small units called “flits” (flow-control units) which are typically one or a few bytes long. A flit can be regarded as the unit of a message which may traverse a channel in one time step. The first flit of a message advances along the route with the remaining flits following in pipelined fashion. This first flit, called the “head”, contains sufficient information for a routing decision to be made. Therefore, only the head is processed at an intermediate node before the message is forwarded. If the head flit blocks due to the unavailability of a channel, the flow control within the network blocks the trailing flits.

Routing is assumed to be deterministic and minimal, i.e., for any source-destination pair, only one route may be used, and it must be the shortest route. Notice that deadlock cycles can occur within a ring due to the wraparound channels. As suggested by Dally and Seitz [3], virtual channels are used to transform these cycles into spirals. In this scheme, all channels in a unidimensional cycle are broken up into pairs of virtual channels (refer to Figure 2). A virtual channel pair is implemented by time-multiplexing onto the associated physical channel. Messages at a node numbered less than their destination are routed on the A virtual channel, while messages at a node numbered greater than their destination are routed on the B virtual channel. Notice that with this routing restriction, any negative-going message which traverses the c_{0-} channel traverses A virtual channels through traversal of $c_{0-,A}$. At this point, the message switches to B virtual channels to complete its route. A similar observation is made for positive-going messages. It is this characteristic which transforms the cycles into spirals and thus makes the algorithm sufficient for deadlock-free routing. The algorithm is extended to support deadlock-free routing in general k -ary n -cubes by coupling it with a dimension ordering.

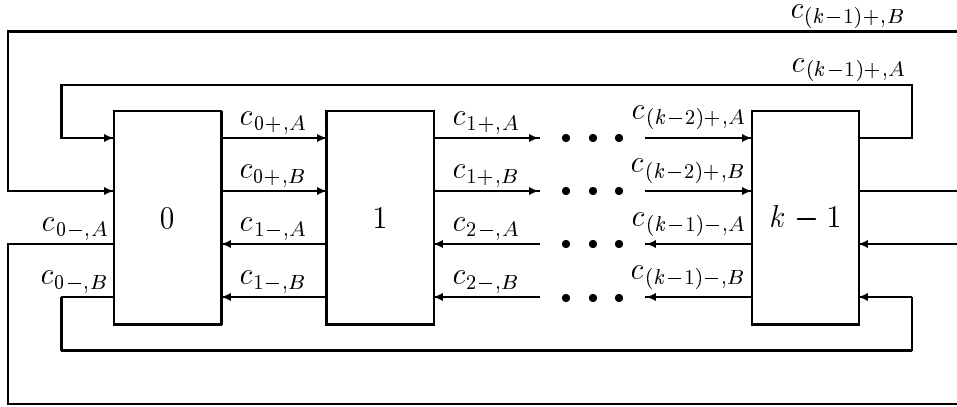


Figure 2: Superposition of virtual channels onto physical channels in a ring

3 The Red Rover Routing Algorithm

As shown in [3], some ordering of resources is necessary to provide deadlock prevention in routing on networks with wraparound connections, e.g., rings. For unidirectional rings, imposing an ordering via the utilization of virtual channels to effectively transform the ring into a spiral is as plausible a solution as any. However, for bidirectional rings, an alternative ordering of the virtual channels is possible. As will be shown, this scheme, which we label the Red Rover Algorithm, still provides deadlock-free routing yet results in better performance characteristics and a simpler hardware implementation.

The Red Rover Algorithm is extremely simple. The network is organized as two equal sets of contiguous nodes, with separation points arbitrarily chosen, e.g., designate nodes 0 through $(k/2 - 1)$ as one group and nodes $k/2$ through $(k - 1)$ as the other. (For rings with an odd number of nodes, one grouping will contain one more node than the other.) Messages sent from one group of nodes are always sent along *A* virtual channels, regardless of destination. Similarly, the other group of nodes always sends messages along *B* virtual channels. For purposes of this paper, we arbitrarily prescribe that nodes 0 through $(k/2 - 1)$ inject on the *A* channels, and nodes $k/2$ through $(k - 1)$ inject on the *B* channels. Furthermore, messages injected on *A* virtual channels continue their traversals on *A* channels until they reach their destinations. Similarly, messages injected on *B* channels are confined to *B* channels for their entire traversals. This bisection effectively forms four disjoint networks: the positive *A* network, the positive *B* network, the negative *A* network, and the negative *B* network.

We merely need to show that no cycles exist in the ordering of virtual channels to prove the Red Rover Algorithm is deadlock-free, where the ordering corresponds to the order in which messages may traverse the channels. Observe the ordering of the positive *A* channels. Notice that no node in the range of 0 through $(k/2 - 1)$ may send a message further than node $(k - 1)$ along these channels (refer to Figure 2). This stipulation results from the minimal routing assumption. Therefore, channel $c_{(k-1)+,A}$ is not used, and the resulting channel ordering is $c_{0+,A} > c_{1+,A} > \dots > c_{(k-2)+,A}$. A similar ordering can be formulated for the positive *B* channels, the negative *A* channels, and the negative *B* channels to show that each ordering contains no cycles. Since a message is confined to one of these sets for its entire traversal, i.e., it may not switch from one set to another at some point during its traversal, the scheme is deadlock-free.

4 Message Traffic Analysis

Before detailing the simulation experiments, some observations of message traffic on a ring demonstrate why improved performance is expected with the Red Rover algorithm. Assume that each node injects messages at a rate of λ . Furthermore, let the destination for each message be selected randomly from all nodes other than the source node. Also, we assume an even number of nodes in the ring to keep the analysis clear and concise. It is a straightforward exercise to extend the analysis to a network containing an odd number of nodes. Lastly, we assume that messages destined for a node which is $k/2$ hops from the source are routed on the negative-going channels. Given these assumptions, we wish to obtain values for the message traffic rate on each virtual channel in the network for both the Red Rover algorithm and the spiral algorithm of Dally and Seitz [3]. Because of the symmetry of the network, it is necessary only to analyze one direction of channels. We therefore confine our analysis to the negative channels and state that similar findings result for the positive channels.

We will first analyze the negative A channels for the Red Rover scheme. First observe the traffic on $c_{0-,A}$ (refer to Figure 2). Note that this channel carries all of the negative-going traffic that emanates from node 0. Since node 0 sends to $k/2$ destinations on negative channels, and there are $k-1$ possible destinations, this portion of traffic amounts to $k\lambda/2(k-1)$. In addition, $c_{0-,A}$ carries all the negative-going traffic sourced from node 1 except that destined for node 0. Thus, node 1 contributes $(k/2-1)\lambda/(k-1)$ to the traffic rate on $c_{0-,A}$. Continuing the analysis in this manner, the traffic rate on $c_{0-,A}$ is calculated as

$$\lambda_{0-,A} = \frac{\sum_{j=1}^{k/2} j}{k-1} \lambda = \frac{k(k+2)}{8(k-1)} \lambda .$$

This method of analysis is used to calculate the message traffic rate on any virtual channel in either the Red Rover or the spiral routing algorithm. Specifically, the traffic rate for a channel is found by identifying which nodes can source messages which will use the channel and then summing the individual traffic rates generated by each of the identified nodes on the channel. For the Red Rover routing algorithm, the traffic rates on the negative channels are specified by

$$\lambda_{i-,A} = \begin{cases} \frac{\sum_{j=i+1}^{k/2} j}{k-1} \lambda = \frac{-4i^2-4i+k^2+2k}{8(k-1)} \lambda & 0 \leq i < k/2 \\ \frac{\sum_{j=0}^{i-k/2} j}{k-1} \lambda = \frac{4i^2+(4-4k)i+k^2-2k}{8(k-1)} \lambda & k/2 \leq i \leq k-1 \end{cases}$$

$$\lambda_{i-,B} = \begin{cases} \frac{\sum_{j=0}^i j}{k-1} \lambda = \frac{i^2+i}{2(k-1)} \lambda & 0 \leq i < k/2 \\ \frac{\sum_{j=i+1-k/2}^{k/2} j}{k-1} \lambda = \frac{-i^2+(k-1)i+k}{2(k-1)} \lambda & k/2 \leq i \leq k-1 \end{cases}$$

Similarly, the traffic rates on the negative channels in the spiral routing algorithm are calculated as

$$\lambda_{i-,A} = \begin{cases} \frac{\sum_{j=0}^{k/2-i} j}{k-1} \lambda = \frac{4i^2-(4+4k)i+k^2+2k}{8(k-1)} \lambda & 0 \leq i < k/2 \\ 0 & k/2 \leq i \leq k-1 \end{cases}$$

$$\lambda_{i-,B} = \begin{cases} \frac{\sum_{j=0}^{k/2} j - \sum_{j=0}^{k/2-i} j}{k-1} \lambda = \frac{-4i^2+(4+4k)i}{8(k-1)} \lambda & 0 \leq i < k/2 \\ \frac{\sum_{j=0}^{k/2} j}{k-1} \lambda = \frac{k^2+2k}{8(k-1)} \lambda & k/2 \leq i \leq k-1 \end{cases}$$

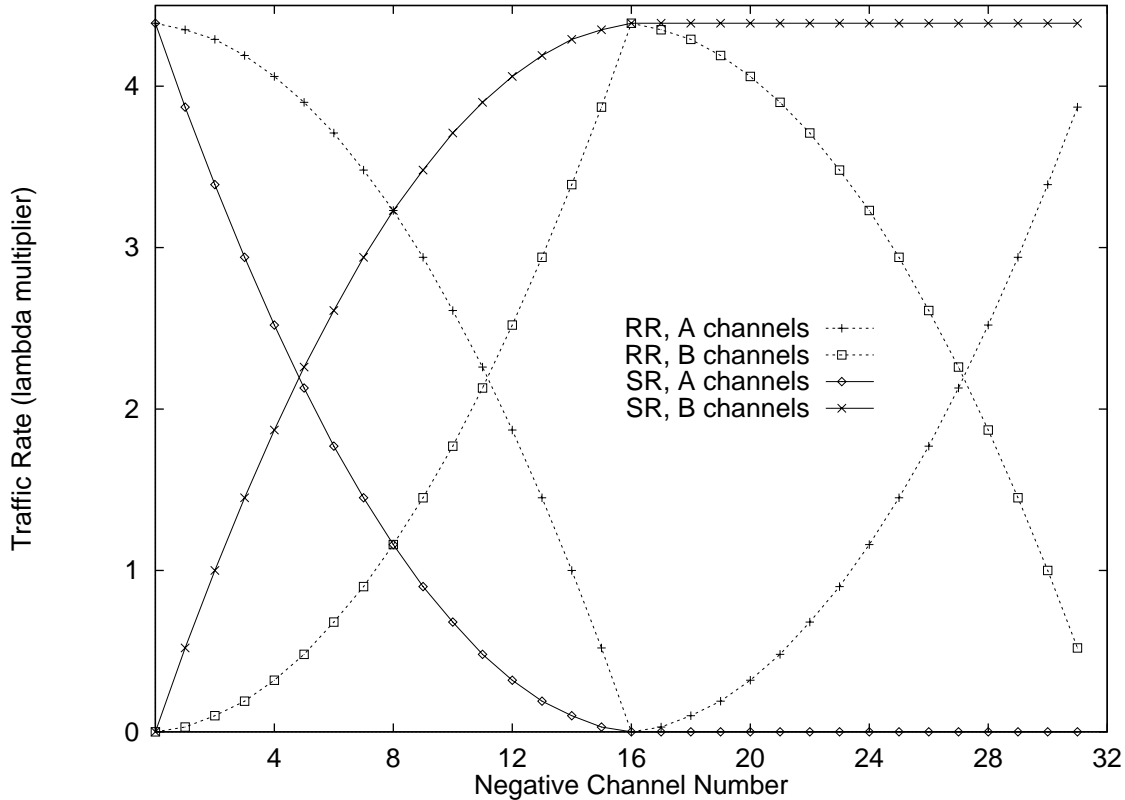


Figure 3: Channel traffic rate for 32-node ring, random traffic

With these equations, it is helpful to plot channel traffic rate as a function of channel number. Plots for each routing algorithm on a 32-node ring are shown in Figure 3. (RR denotes Red Rover routing, and SR denotes spiral routing.) A couple of observations indicate that the Red Rover algorithm offers improved performance in terms of average message latency. First notice that the Red Rover algorithm does a better job of utilizing available resources in that only two virtual channels carry no traffic, $c_{0-,B}$ and $c_{16-,A}$. In contrast, one quarter of the virtual channels are unused in the spiral algorithm. Thus, the capability to multiplex virtual channels on a physical channel is better exploited by the Red Rover algorithm. This property is especially critical because virtual channels help alleviate the blocking penalty of wormhole routing by providing the means for messages to bypass blocked messages.

Another observation is that the most heavily-used channels are adjacent to each other in the spiral routing scheme. As quantified in [5], the pipelined and blocking nature of wormhole routing causes the waiting times of channels to increase the waiting times of preceding channels. Although the amount of channel interaction is limited by the distance between the channels and message length, the Red Rover algorithm fares better in the realm of cascaded blocking penalties because for most sets of contiguous channels, the average channel traffic load is lower than that of the spiral routing algorithm.

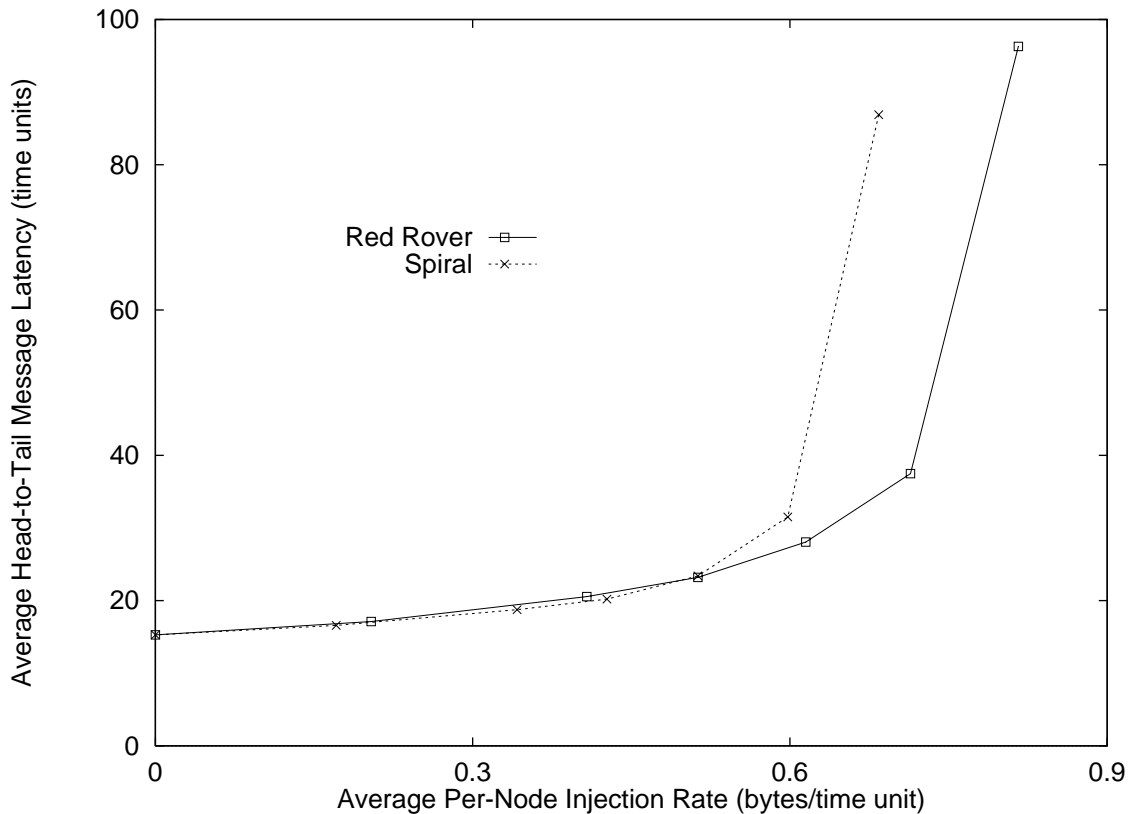


Figure 4: 16-node ring, 32-bit channels, 40-byte messages, random traffic

5 Simulation Experiments

The preceding channel traffic analysis indicates that for random traffic, the Red Rover algorithm provides lower message latencies than the spiral scheme. To validate this prediction, we conducted simulation experiments. The simulator is a C++ program that simulates message transfers in a k -node bidirectional ring at the flit level. A simulation time unit (cycle) is defined as the time required for a flit transfer on a physical channel. Virtual channels are modeled fairly in that the flit of only one virtual channel may be allocated the physical channel during one time unit. The simulator allows several input parameters: k , traffic rate, message length, routing algorithm selection, etc.

For all experiments presented, the simulator was run with random destination traffic patterns. The results presented are the averages of several runs. For each run, statistics gathering was initiated only after warm-up transient effects became negligible. Experiments were conducted for 16-node and 64-node rings with 40-byte messages. These values are representative of current multicomputers, which rarely contain more than 64 nodes along a dimension. Also, a typical message is a cache line with some header information (a total of about 40 bytes). For another data point, we also simulated 256-byte messages on the 16-node ring. Based on projections for next-generation multicomputers, the channel width for all cases was assumed to be 32 bits. The results for average head-to-tail message latency versus traffic load for these cases are presented in Figures 4 through 6. Head-to-tail latency is defined as the amount of time from the generation of a message until the tail flit of the message is received at the destination. For all results shown, the Red Rover algorithm offers lower message latencies at high

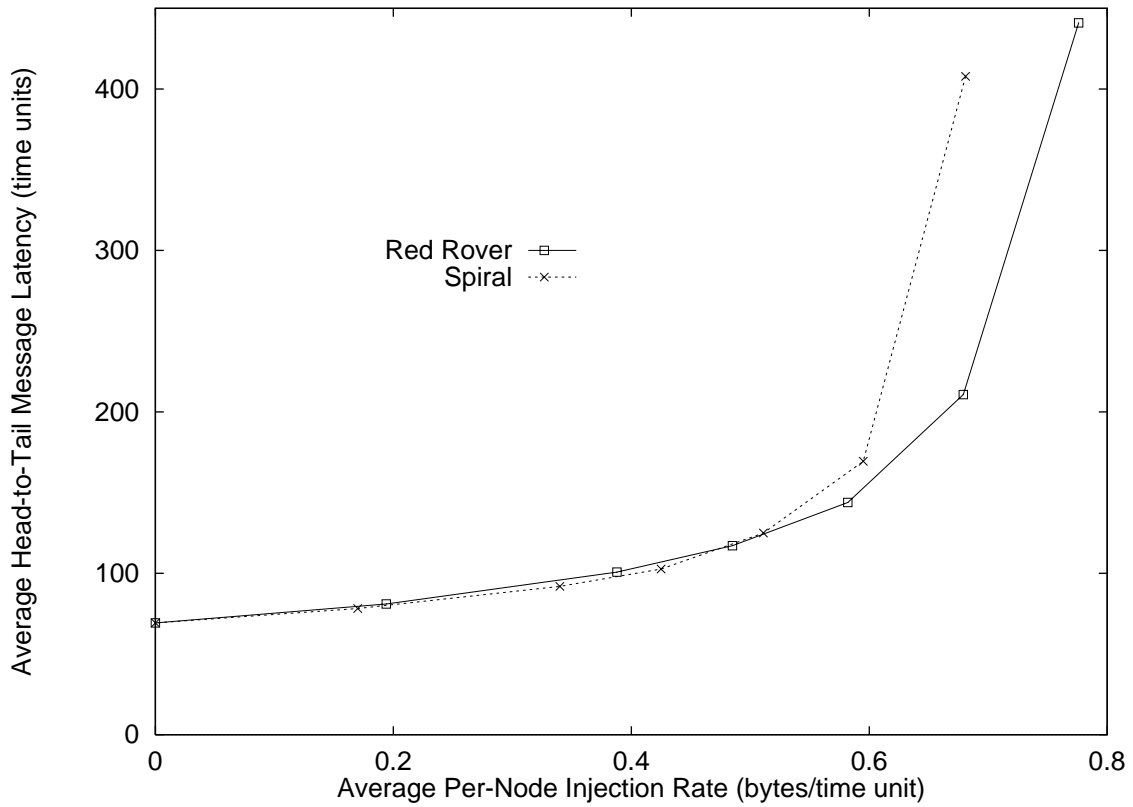


Figure 5: 16-node ring, 32-bit channels, 256-byte messages, random traffic

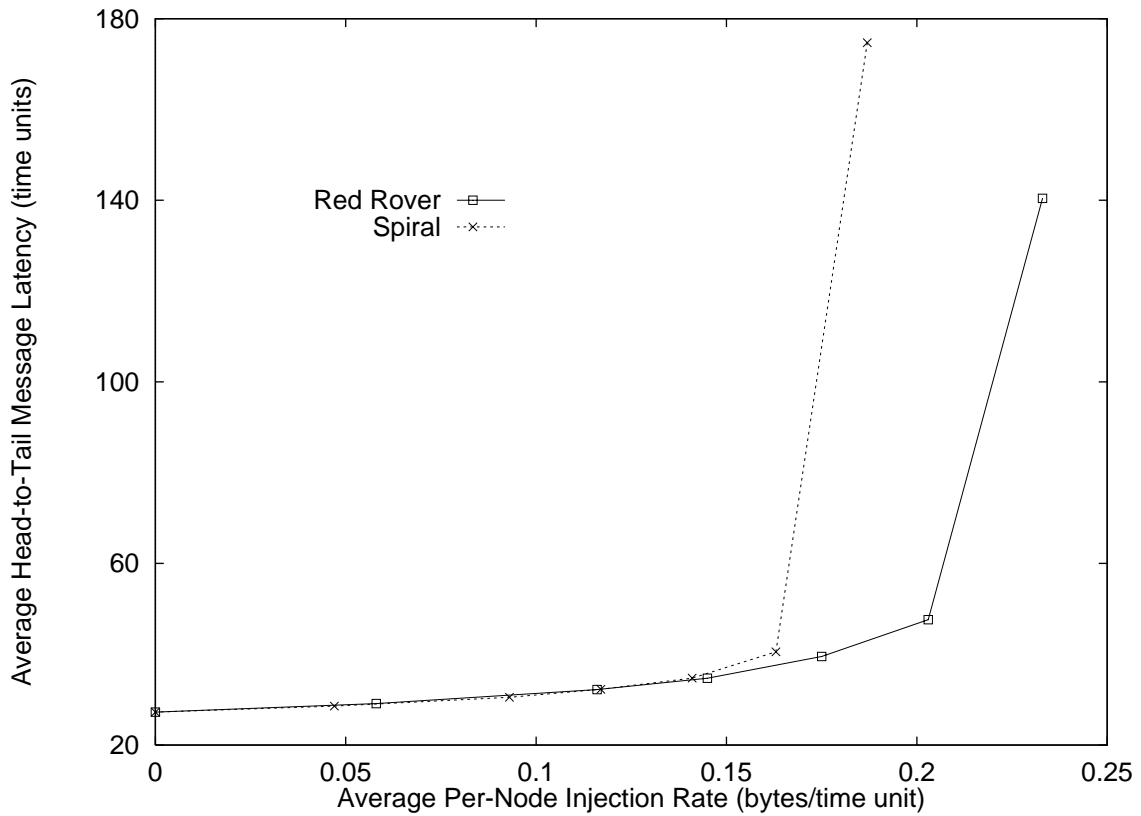


Figure 6: 64-node ring, 32-bit channels, 40-byte messages, random traffic

traffic rates. Furthermore, the achievable throughput for the Red Rover scheme is 20 to 25% higher than that of the spiral method (indicated by the vertical asymptotes of the curves). Other experiments yielded similar results.

6 Comments

In addition to improved performance, the implementation of a Red Rover routing element is simple and “intuitively cleaner” than that of a spiral router. For the spiral router, two decisions must be made for a message to be routed from the injection channel to an output channel (refer to Figure 1). The direction, + or -, and the virtual channel, A or B , must be determined. Since the virtual channel chosen is generally dependent on both the source and destination, this decision must be made on the fly. In contrast, the virtual channel decision for the Red Rover router can be hardwired from either a control register or an input to the router since this choice is dependent only on the source. In other words, the switching from the injection channel to output channels for the Red Rover router requires less logic and is thus faster.

This reasoning also applies to switching between network input channels and network output channels. Since a message can switch from A virtual channels to B virtual channels in the spiral algorithm, a decision element is required. Although this decision can be hardwired because this switching always takes place at the same node, it still requires extra resources that the Red Rover does not. This characteristic also necessitates that the spiral router use 3-way splitters on the input A channel datapaths and 3-way mergers on the output B channel datapaths. Since the Red Rover router requires only 2-way splitters and mergers on these datapaths, it has a smaller implementation, all other factors being equal.

Lastly, we discuss switching from network input channels to the ejection channel. In general, the spiral router must be capable of switching from both A and B channels to the ejection channel, although this decision can be hardwired since it depends only on the destination node number. The Red Rover router must make this decision on the fly. Despite this one drawback, the Red Rover router is clearly more area-efficient and faster than the spiral router. Since a “time unit,” or cycle time, for a Red Rover router is less than that for a spiral router, the relative improvement in latency is even greater than that exhibited in the simulation results of Section 5.

One disadvantage of using the Red Rover scheme is the lack of fault-tolerance. The Red Rover algorithm allows only one path for a given source-destination pair, the shortest path. Indeed, allowing the other path as well would cause the algorithm to lose its property of deadlock prevention. In contrast, the spiral algorithm allows both possible paths for a source-destination pair and still maintains its deadlock-free characteristic.

7 Conclusion

The Red Rover algorithm, a deadlock-free routing scheme for bidirectional rings, has been presented. By coupling this scheme with a dimension ordering, the algorithm extends to general k -ary n -cubes. This algorithm is better at distributing the message

traffic among virtual channels than the spiral algorithm of [3]. It thus makes the routing network more symmetric, causing it to provide lower message latencies and higher throughput than the spiral algorithm. This property has been indicated by a message traffic analysis and validated through simulation experiments on a variety of network sizes and message lengths. Furthermore, the implementation of a Red Rover routing element is less complex than that of a spiral router. Thus, the Red Rover router is both smaller and faster. The only drawback to the Red Rover scheme is that it allows only one path from source to destination, while the spiral scheme allows two paths. However, most current multicomputers, such as the Intel Paragon, use deterministic routing hardware and can only support fault-tolerant routing by relaying messages through intermediate nodes. The Red Rover algorithm does not preclude this type of fault-tolerant routing. Therefore, the Red Rover routing scheme keeps pace with current trends in the area of fault-tolerance while improving performance and VLSI implementation.

Acknowledgments

This research was supported by ARPA under contract J-FBI-91-282.

References

- [1] W.C. Athas and C.L. Seitz, Multicomputers: Message-passing concurrent computers. *Computer*, August 1988, pp. 9–25.
- [2] Cray Research, The Cray T3D System. World-Wide Web, http://www.cray.com/PUBLIC/product-info/mpp/CRAY_T3D.html.
- [3] W.J. Dally and C.L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, May 1987, pp. 547–53.
- [4] W.J. Dally, *A VLSI Architecture for Concurrent Data Structures*. Kluwer, 1987.
- [5] J.T. Draper and J. Ghosh, A comprehensive analytical model for wormhole routing in multicomputer systems. *Journal of Parallel and Distributed Computing*, November 1994, pp. 202–14.
- [6] Intel Scalable Systems Division, Intel Paragon Supercomputers. World-Wide Web, <http://www.ssd.intel.com/paragon.html>.
- [7] Kendall Square Research, KSR Hypertext Library. World-Wide Web, <http://www.ksr.com/>.
- [8] P. Kermani and L. Kleinrock, Virtual cut-through: A new computer communication switching technique. *Computer Networks*, March 1979, pp. 267–86.
- [9] L.M. Ni and P.K. McKinley, A survey of wormhole routing techniques in direct networks. *Computer*, February 1993, pp. 62–76.
- [10] C.L. Seitz, The Cosmic Cube. *Communications of the ACM*, January 1985, pp. 22–33.