Adaptive Routing in RS/6000 SP-like Bidirectional Multistage Interconnection Networks

Mohammad Banikazemi†1 Craig B. Stunkel‡
Dhabaleswar K. Panda†2 Bulent Abali‡3

†Dept. of Computer and Information Science
The Ohio State University
Columbus, OH 43210
email:{banikaze, panda}@cis.ohio-state.edu

‡System Design & Performance
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
email:{stunkel, abali}@watson.ibm.com

Abstract

The IBM RS/6000 SP is one of the most successful commercially available multicomputers. SP owes its success partially to the scalable, high bandwidth, low latency network. In this paper, we present the adaptive routing scheme used in the new SP network switch chip called the Switch2[4]*. We show that the adaptive routing methods outperform the oblivious routing methods on SP like multistage networks. It is shown that the adaptive routing increases the network throughput by up to 229% over oblivious routing in some cases. We also study the effect of output selection functions on the network performance. We present six different output selection functions and study their performance for different system parameters and communication patterns through extensive simulation. The results show that three of these selection functions perform similar to each other and outperform the other selection functions consistently. Therefore, unlike previous research findings for meshes and tori [9], there is no need to use multiple (or hybrid) selection functions to obtain the best performance for the SP like bidirectional multistage interconnection networks. We also provide an analysis of the cost-effectiveness of different selection functions with respect to the complexity of their hardware implementations.*

1 Introduction

A high performance interconnection network [4] is a crucial component in multicomputer and clustered systems today. Parallel and distributed applications depend on interconnection networks for achieving good overall performance. One of the keys to achieving good network performance is the routing decisions made within the network. Routing decisions can be classified as adaptive or oblivious [8]. In adaptive routing schemes, switching elements choose a route among several options and the route selection is done dynamically depending on the network traffic. In the oblivious routing scheme a fixed routing decision–oblivious to the traffic is made. Thus, a message can use only a predetermined single path between the source and destination nodes. For these reasons adaptive routing networks perform better than oblivious routing networks in many cases. This paper focuses on the adaptive routing features of the recently developed Switch2 chip which may be used in future IBM RS/6000 SP networks.

The RS/6000 SP system is one of the most successful parallel systems commercially available today. The SP system's success is partly due to the scalable, high bandwidth, low latency SP interconnect. The recently developed Switch2 chip [16] has many enhancements over its predecessors, including higher link rate of 500 MBytes/s, larger internal buffers, faster cycle time, and support for longer links. However, one of the most important architectural enhancement of the Switch2 chip is the support for adaptive routing function. A hardware multicast support has also been added to Switch2 although it will not be discussed in this paper.

In adaptive routing, a switching element (e.g. Switch2) examines the header of an incoming message and may determine that more than one output port are available for forwarding the message to its destination. The switching element uses an *output selection function* to decide which output port will be used [5, 8]. It has been shown that using a proper output selection function is important for achieving good adaptive routing performance [7, 9, 13]. In this paper, we focus particularly on the output selection functions for SP-like, Bidirectional Multistage Interconnection Networks (BMIN). BMINs have aggregate bandwidth scaling linearly as the number of nodes increases, as do unidirectional multistage networks. They have smaller diameter than the mesh and torus networks (e.g. $O(logN)$ vs. $O(N^{0.5})$). Fat trees [11] and least common ancestor networks [12] are examples of BMINs.

Previous studies on output selection functions have focused on the mesh and torus networks [3, 7, 9, 10, 13, 20, 21]. To the best of our knowledge, output selection functions for BMINs or MINs have not been studied to this date. In this

[1]This author's work is supported in part by an IBM Cooperative Fellowship award. Work presented in this paper was performed while visiting the IBM T. J. Watson Research Center.

[2]This author's work is supported in part by an NSF Career Award MIP-9502294, NSF Grant CCR-9704512, and an Ameritech Faculty Fellowship award.

[3]To whom all correspondence should be addressed.

[4]Note that Switch2 is an experimental technology. This is not a product announcement. Furthermore, no assumptions should be made regarding availability of Switch2 as a product in the future. Switch2 is not an IBM internal or external or a product name. It is used as a place holder in this paper.

paper, we focus on this challenge. Based on the architectural characteristics of the Switch2 switch chip and SP-like BMINs, we introduce six output selection functions with different hardware complexities. These functions are: LRU, MRU, RND, RR, LRUC, and LRUD. The performance of adaptive routing on SP-like BMINs with the proposed output selection functions is studied with extensive simulation. A set of different traffic patterns, different network sizes, and a range of message lengths are used in the simulation experiments. Simulation results reveal many interesting properties of adaptive routing on BMINs. The main contributions of this paper are:

1) We describe the adaptive source routing architecture of the recently developed Switch2 chip. Switch2 combines adaptive routing and source routing which is a unique feature. Adaptive source routing method allows a source node to send messages in fully or partially adaptive or in oblivious fashion on a per packet basis. Performance of oblivious routing and adaptive routing network are compared by simulations.

2) We propose three variations of the LRU output selection function: per input least recently used (LRU), centralized least recently used (LRUC), and destination based least recently used (LRUD). These functions attempt to capture and classify output selection history of previous messages and then make future output decisions based on this information. LRU captures information in a per input port basis. LRUC captures information in a per switch chip basis by taking information from all input ports. LRUD captures information from all input ports and classifies them according to their destinations, hence keeping more detailed information. We discover through simulations that LRUC performs worse than LRU. This result is totally unexpected and counterintuitive. We also show that the LRUD performs marginally better than LRU. We find no significant performance difference between LRU and RR (round robin) output selection functions. We also show that the MRU (per input most recently used) and RND (random) output selection functions do not perform well. These results and observations provide us guidelines to design a switch with the best selection function.

3) We observe that good performing selection functions must minimize contention in the downward path from the least common ancestor (LCA) switches (of source–destination node pair) to the destination node. Because when adaptive routing is used in BMINs, contention occurs only in the downward path from an LCA switch to the destination node where no adaptive choices are left, hence message must follow a fixed route. The LRU and RR selection functions accomplish this objective in a probabilistic manner by distributing the messages throughout the upward paths to the LCAs. The proposed destination based LRU (LRUD) output selection function attempts to accomplish this objective in a more deterministic manner than the LRU selection function, however with increased hardware complexity.

4) We discover that the best selection functions for BMINs are not dependent on the traffic pattern, message size, or system size. Therefore in BMINs there is no need to implement (in hardware) multiple selection functions for different classes of traffic. This is contrary to the findings of other researchers [9] which suggest that no single selection function works best for all communication patterns in meshes and tori.

5) We discuss the hardware complexity of implementing the output selection functions and show which selection function provides a good cost/performance ratio.

6) We find no oblivious routing scheme that performs significantly better than adaptive routing in BMINs. This is unlike the findings in [9] which indicates that oblivious routing works better than adaptive routing in the mesh and torus networks for some types of traffic.

The rest of this paper is organized as follows: In Section 2, we give an overview of the SP switch. The basic routing issues and the SP topology are discussed in Section 3. In Section 4, we present the new adaptive source routing architecture of the Switch2 chip and describe the candidate output selection functions for adaptive routing. In Section 5, we present the performance evaluation results and provide a discussion on the performance of different selection functions. Related work is discussed in Section 6. In Section 7, we present our conclusions.

2 An Overview of the SP Switch

Before discussing the adaptive routing function of the Switch2 chip, we present important features of the current SP network products. The current generation of SP networks are built with a set of SP Switch building blocks. The current generation SP Switch building block is an 8×8 switch chip that achieves link bandwidths of 150 MB/s/link/direction, and this switch chip is architecturally similar to the 40 MB/s Vulcan switch chip [17] used in the original IBM SP1 [1, 18] and SP2 [19] High Performance Switch networks. The basic organization of an SP switch chip is shown in Fig. 1.

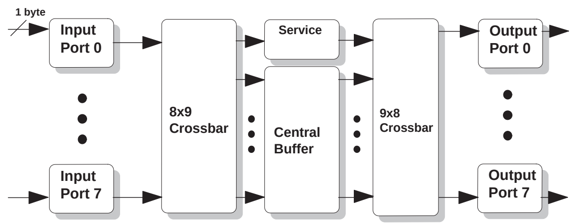

Figure 1: The SP switch chip organization.

Current SP Switching network products use wormhole routing [6] which consists of both flit-based flow-control and cut-through switching. In flit-based flow-control, a packet is broken into small units called flits, or flow-control digits. A flit is the smallest unit of a packet that can be accepted or rejected by the flow-control mechanism. In cut-through switching, once the packet header (which contains the route) is received, the route is decoded, and an immediate attempt is made to forward the packet to the desired output port through the main crossbar. In SP systems, if this attempt fails because the output port is busy, then the switch chip instead begins to transfer the packet to the central buffer (Fig. 1).

In most wormhole implementations, when a packet cannot be routed to the desired output port, it is blocked in place, immediately resulting in head-of-the-line blocking, which prevents any subsequent packet at that input port from being forwarded. However SP switches contain a large central buffer in which packet flits can be stored when an output port is busy. In most cases, the packet can be stored in this central buffer. This frees a subsequent packet at that input port to continue to its destination provided it exits from another output port. This technique is named buffered wormhole routing [19]. The central buffer space is dynamically shared according to demand from the input ports, which increases maximum utilization in the switching element.

When an input port receives a flit, it stores it in the input buffer. Concurrently, the input port may also be fetching flits out of this buffer. When a packet header is fetched from the input buffer, the route is decoded and a crossbar request is made to the indicated output port. Each idle output port which receives crossbar requests grants one of the requests on an individual LRU basis for fairness. If an input port receives a grant from the output port, the packet header is immediately sent through the main crossbar to the output port, and the rest of the packet is also sent through the crossbar in a pipelined fashion.

3 Routing and Topology

In the current and previous generations of SP switch chips, the switches are source-routed [16], which means that the routing decision for each switch chip is embedded within the message packet. Each switch chip decodes and then discards the first route field in the packet. Each route field contains a 3-bit encoded output port ID. Source routing allows different packets going to the same destination to traverse different paths based upon the embedded route. It is straightforward to bypass faulty links or switches using source routing, and, in principle, no switch chip setup is required before sending a source-routed packet through that switch chip.

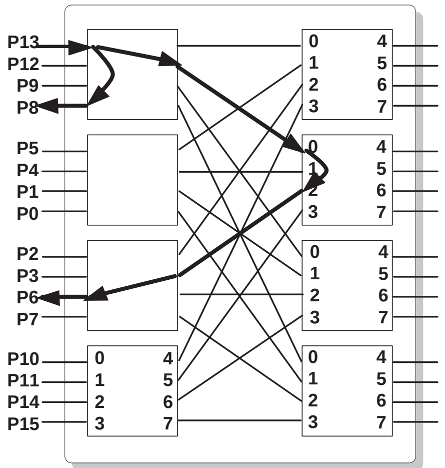

Figure 2: A 16-node SP switching network.

Even though the source routing technique assumes no particular topology, certain topologies are more scalable in aggregate bandwidth than others. Because SP systems are positioned to be highly scalable, the choice of topology is critical. From the earliest SP1 machines, the IBM SP systems have been built with only bidirectional multistage interconnection networks (BMINs). An example of a BMIN is the 16-node SP network shown in Fig. 2, where each block represents a switch chip in the network. Similar to the uni-directional multistage networks, these networks scale aggregate bandwidth linearly as the number of nodes increases. BMINs reward communication locality, and require no virtual channels to avoid deadlock among packets in the network. Fat-trees [11] and least common ancestor networks [12] are examples of BMINs. Figure 2 shows the path of two packets sent from node P13. The first message is destined for P8, and need only be routed through one switch chip. The second message is destined for P6, and traverses three switch chips. Note that any of the four right-hand-side switch chips could have served as the second switch chip in the path of this packet. In all cases a minimal (shortest-path) route is selected. Note that the node numbering in Fig. 2 is same as the numbering used in 16 node SP systems and it is an artifact of the wiring layout.

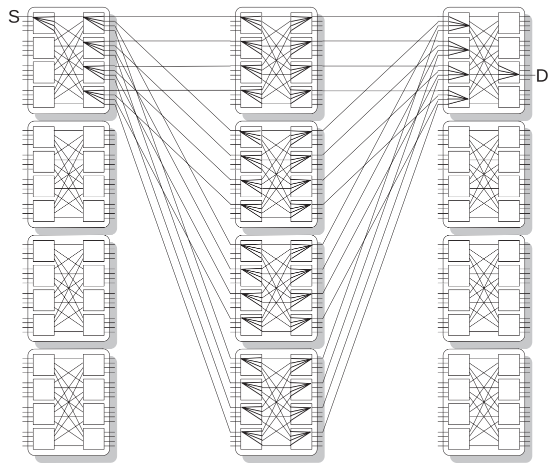

Figure 3: A 128-node SP switching network. A subset of the entire set of links is shown.

Another SP network example for 128 nodes is shown in Fig. 3. This network consists of two sets of 64 nodes on the left-hand-side and right-hand-side of the network, respectively. Eight node switch boards on both sides connect to 128 nodes, and four intermediate switch boards in the middle interconnect left-hand-side and right-hand-side node switch boards. Figure 3 illustrates only a subset of the network links for simplicity. Missing links follow the same wiring pattern between the node switch boards and intermediate switch boards.

Current generation SP networks use non-adaptive (commonly known as oblivious) routing. In oblivious routing, the routing decision for each switch chip is made apriori, and in the source routing method these decisions are embedded in the message packet header by the source node. In richly interconnected networks many routes may exist between a pair of source and destination nodes. For example, in Fig. 2 there are four minimal routes from P13 to P6, and in Fig. 3 there are 64 minimal routes from node S to D. Therefore, an important issue in oblivious routing is to decide which of the multiple routes to use for a given packet. Current generation SP networks use a method known as *4-route oblivious routing.* In this method, on each node, the network interface maintains a route table containing four separate routes per destination node. The four routes are generally different. Each node sends consecutive messages destined to a node through these four routes in a round–robin fashion. The objective of this scheme is to spread message traffic as much as possible across the network and eliminate potential “hot-spots.” The 4-routes per source–destination pair are selected by a static load balancing algorithm that attempts to distribute the aggregate network traffic evenly across all the links [1]. More than four routes per destination could have yielded better performance. However, a tradeoff was made to minimize memory requirements of the route table in the network interface adapter [1].

4 Adaptive Routing and Output Selection Function

In this section, we first discuss the architectural support for adaptive source routing in Switch2 switches. Then, we explain how the routing information for adaptive source routing is embedded in packets. We also discuss how six different output selection functions perform the selection operation.

4.1 Adaptive Source Routing

One of the most important architectural enhancement in the Switch2 chip is the adaptive source routing. Multicast routing function was also added to Switch2 but will not be discussed in this paper [16]. We should emphasize that although the adaptive routing capability exists in the Switch2 chip, the use of it depends on the software that controls the route format at the network interface. Any product level use of this function might be phased in over time as experience is gained in its application.

In non-adaptive (or oblivious) routing, a fixed routing decision is made for traveling between a source and a destination node. Each switch must forward the message through a fixed output port regardless of the traffic conditions. For example if the pre-determined output port is busy, the packet will wait until that port becomes available, although there may be other ports available in the switch. In contrast, adaptive routing methods allow for more than one choice of output ports and routing decisions are made by the switches. A switching element may forward the packet to one of many output ports making the routing decision on the fly as the packet header is decoded. For example, it may forward the packet to an available output port or a less busy output port if several are available.

While the Switch2 chip supports adaptive routing, for many reasons including backward compatibility, Switch2 must also continue to support oblivious routing and even a mix of oblivious and adaptive routing. Existing adaptive routing schemes rely on distributed routing techniques (destination routing) as opposed to source routing. For example, in a regular 2-D mesh, an intermediate switch can, based upon the destination address or a distance vector from the destination, choose from more than one output port that will lead to that destination. Fault-tolerance becomes an issue for such networks, as faulty links make it possible to select a link that eventually leads to a dead-end. This problem is sometimes addressed by adding virtual channels for adaptive routing and/or by allowing non-minimal routing. A more powerful fault-tolerance capability can be achieved though a form of destination routing known as table-lookup routing, in which each packet carries a logical address that is used to index a route lookup table inside each switch in the path. The tables throughout the network can be configured to avoid dead-ends. However, in all of these schemes, the source processor has no per-packet control over the route or the amount of adaptivity allowed for an individual packet. Instead, the routing decisions are made by the switches distributed throughout the network. For SP systems, we desire to maintain control over adaptivity at the source node, and to use minimal routing without any need for virtual channels. For instance, in-order packet arrival may be required in some cases but not others. All of these objectives are met by adaptive source routing when used in conjunction with BMINs [2].

4.2 Route Specification

In adaptive source routing, just as in the current SP source routing technique, the source node embeds a single route field for each switch chip to be traversed in the path. However, for adaptive route fields, a 4-bit field is used to specify the permissible set of output ports instead of a single output port. To see why this is sufficient, recall the SP BMIN topology shown in Fig. 2. Each packet in an SP network traverses a minimal path, traveling away from the source node to some least common ancestor (LCA) of the source and destination nodes. (In Fig. 2, for the packet traveling to P6, any of the right-hand-side switches is an LCA of the P13-P6 pair of nodes.) The packet then travels downward to the destination. To maintain a minimal route, there is only one path downward from an LCA. Thus all adaptivity must occur on the upward path. Each routing decision along the upward path involves at most the 4 output ports on the upper side of the switch chip. As shown in Fig. 4, the entrance port guides the interpretation of the route field. On the upward path to LCA, a packet entering the switch chip from one side must always exit from the other side. Therefore, the adaptive routing nibble needs to specify only 4 bits each corresponding to an output port on the opposite side of the input port.

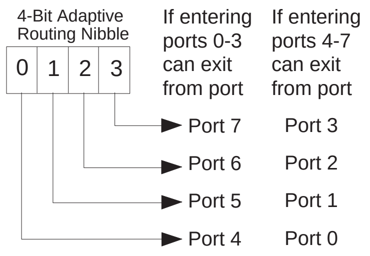
4-Bit Adaptive Routing Nibble
0
1
2
3
If entering ports 0-3 can exit from port
If entering ports 4-7 can exit from port
Port 7
Port 3
Port 6
Port 2
Port 5
Port 1
Port 4
Port 0

Figure 4: Adaptive routing nibble bits set to ’1’ indicate which ports a message can exit from.

Because there is only one route field per switch stage in the message header, any LCA switch reachable via the adaptive choices must have the same downward view of the destinations. Fortunately, this is an inherent property of fat-tree networks, and in most current SP networks this is also the case. For instance, to send a packet from P6 to P13, the same port (port 2) of each LCA switch leads to the destination (Fig. 2). There are SP network topologies in which only a subset of LCAs satisfy this property. For these networks, the adaptivity on the upward path can be restricted with adaptive source routing so that only this subset of LCA switches will be reached. Note that a 4-bit adaptive route field is not sufficient to determine the LCA switch (the point at which the packet must turn and continue non-adaptively). Therefore Switch2 packets also carry an initial field in the first flit which maintains the number of remaining adaptive routes in the packet. All remaining route fields in the packets are non-adaptive, and carry the same format as previous generations of SP switch chips. This has the added advantage that the routes generated for older SP systems can be trivially embedded in Switch2 packets.

For example, consider the 128 node network in Fig. 3. From source S to destination D there are 64 different routes out of which a message may follow one route to avoid network congestion. In the first three stages of switches (in the upward path to the LCAs) a message has a choice of four output ports per stage thereby having a total choice of $4 \times 4 \times 4 = 64$ different routes. The first three stage switches will adaptively decide on one of the 64 routes. Once the message moves on to the fourth stage (in the downward path from LCA), it has no choices left but must exit from a fixed port in each stage. Therefore, in overall there are a total of 64 possible routes from node S to D. The example in Fig. 3 illustrates the power of adaptive routing in BMIN where a single oblivious route is replaced by 64 possible routes, one of which needs to be selected dynamically depending on the traffic.

In Fig. 3 since there are six stages of switch chips from S to D, the 4-bit routing fields in the message packet will be in the form of $A, R1, R2, R3,\ R4, R5, R6$ where $A = 3$ indicates that there are three adaptive routing fields followed by the non-adaptive routing fields. Each switch chip decrements A while consuming the first adaptive routing field. The source node may set a field as $R = 1111$ instructing the corresponding switch that any one of the four output ports may be used. For *maximum* adaptivity, hence performance, the first three route fields can be set to $R1 = 1111$, $R2 = 1111$, and $R3 = 1111$. Alternatively, the source node S may send the message *partially* adaptive by turning off some bits in $R1, R2$, or $R3$. The source node may also send the packet obliviously by setting only one bit in each of the three adaptive fields. Thus, the number of distinct routes a packet may follow is:

$$N_{\text{routes}} = |R_1| \times |R_2| \times \cdots\cdots \times |R_{n-1}| \times |R_n|$$

where $|R_i|$ is defined as the number bits set in the routing field R_i.

Being able to specify the degree of adaptivity on a per packet basis and at the source node is one of the unique advantages of the Switch2 chip. This architecture is not only backward compatible with previous generations of SP networks but it also allows a mix of adaptive and oblivious traffic to coexist in the same network. For example, for in-order-delivery, consecutive packets may be sent obliviously through a single route ($N_{\text{routes}} = 1$), while other packets not sensitive to in-order delivery may be sent with maximum adaptivity ($N_{\text{routes}} = \max$). For security or partitioning purposes, packets may be sent partially adaptive to avoid certain regions of the network ($1 < N_{\text{routes}} < \max$).

4.3 Output Selection Function in Adaptive Routing

In the Switch2 chip when an input port decodes an adaptive route field, if more than one output port is a candidate, the input port requests service from all of these output ports. All free output ports will grant the service request. If one or more grants are received from the output ports, the input port selects a winning output port among granted ports and immediately begins forwarding data to that output port through the crossbar. The losing output ports are notified that they are free to grant requests from other input ports. If no grants are received, the packet is routed to the central buffer, just as for non-adaptive packets. The Switch2 chip implements the LRU output selection function described below. We also discuss here several alternative selection functions that we considered and simulated.

Random (RND) Selection Function: One output port is randomly selected from the set of output ports which have granted input port's service request.

Least Recently Used (LRU) Selection Function: In the switch chip, every input port keeps an ordered list of all the output ports. For example, in the 8 port Switch2 chip there are 8 lists. Each list is kept ordered based on the last time an output port is used by a message entering from this input port. After the input port makes a service request, among all of the granted output ports, the least recently used output port wins the selection. The motivation for LRU selection function is to distribute message traffic evenly among all output ports.

Most Recently Used (MRU) Selection Function: This selection function is similar to the LRU selection function except that the list is ordered in reverse. After the input port makes a service request, among all the granted output ports, the MRU port wins the selection. The motivation for MRU selection function is to see if multi–packet messages will perform better if they are concentrated into a single output port in order to avoid mixing with other traffic throughout the network.

Round Robin (RR) Selection Function: In the 8-port switch chip, each input port has a 3-bit register called RRID, which holds the ID (ID=0...7) of the most recently used output port used by a message entering from the input port. After the input port makes a service request, one or more output ports will grant the request. The RR selection function is such that the RRID register is incremented (modulo 8) until a granted output port is found whose ID matches the RRID register. Thus, granted outputs are considered in round robin fashion starting with the last *output port number + 1 (mod 8)*. Note that the RR function is not same as the LRU function. Because LRU keeps a history of all output ports and favors un-granted (busy) output ports by increasing their priority, whereas RR doesn't keep any history except for the most recently output port ID.

Centralized Least Recently Used (LRUC) Selection Function: LRUC is similar to the LRU selection function. The difference is that in the LRUC selection function there is only one LRU list *per switch chip*, while the LRU selection function uses one LRU list *per input port.* The motivation behind the LRUC function is to make the input ports share one LRU list thereby making a switch-chip-wide decision when balancing the output traffic. In contrast, in the LRU selection function described above, the input ports do not share their LRU information. For example, an output port may appear as the least recently used output port in one input's LRU list and it may appear as the most recently used output another input's LRU list. To fix this seemingly contradictory situation we consider the LRUC selection function.

Destination based Least Recently Used (LRUD) Selection Function: Similar to the LRU selection functions, LRUD uses multiple LRU lists. However, in every switch chip there is one LRU list per *node switch* in the network. A node switch refers to a switch chip which is connected to one or more (usually four) nodes. For example, in Fig. 2 there are 4 node switches connecting to the nodes P0–P15 on the left-hand-side of the network and in Fig. 3 there are 32 node switches connecting to 128 nodes on both sides of the network.

The motivation behind the LRUD selection function is to make a network-wide decision when selecting output ports in a switch chip. The LRUD function attempts to keep separate LRU history of messages going to the same destination in the network. This permits consecutive incoming messages going to different destinations to use the same output port. Since consecutive messages will use separate downward paths from LCAs, even if the first message is blocked downstream, the next message(s) can continue because they will be using different output ports in the LCA switches. For example, in Fig. 2 consider the message path from P13 to P6 which uses output port 2 in the LCA switch. Consider another message path from P12 to P15 using the same LCA switch, however using output port 3. Therefore they use separate ports in the LCA switch and due to the central buffer in the SP switches, there is a high probability that the second message can continue even if the first message is blocked on port 2.

5 Performance Evaluation

In this section, we first give an overview of our simulation setup. Then, we compare the performance of oblivious routing schemes with that of the adaptive source routing. Then, we evaluate the performance of the adaptive routing with different output selection functions. We provide a comprehensive discussion about the effect of output selection functions on the network performance. We also provide a comparison of the hardware complexity of these selection functions and show which selection function is the most cost–effective function.

5.1 Simulation Setup

To evaluate the performance of different routing schemes, we conducted network simulations. We used a flit level simulator based upon a C++ model of SP-like switches. The simulated switches implement *buffered wormhole routing* [19] for flow-control and contain a 4 KB dynamically-shared central buffer.

All simulations assume an open network model containing idealized processor nodes: the nodes contain an infinite transmit queue buffer, and packet flits are immediately pulled from the network as they arrive. We assume an exponential distribution for message injection time (message arrival time). We apply a range of loading to the network, in increments of 0.1, where a load of 1.0 indicates that each node is injecting packets in the network at the maximum link data rate. Latency results include input queuing time and are not shown after saturation (steady-state latency is infinite after saturation, assuming infinite input queues). The maximum packet size is 1 KB, and messages longer than 1 KB are broken into multiple packets before transmission.

The open network model makes it possible to "stress" the network to a far greater degree and cause more contention than might be possible in a real environment. For instance, in the SP systems, the processor software and the network interface hardware control the injection of message packets via strategies such as end-to-end flow control and message interleaving that significantly reduce the possibility of network saturation and the creation of "hot-spots" [15]. However, the open network model simplifies analysis by removing the complex software and network interface factors, and makes it possible to examine a single issue: the contribution of the output selection function to the performance of adaptive networks.

We simulated two oblivious routing schemes: the one-route oblivious routing (O1) and the four-route oblivious routing (O4) as described in Section 3. In the O1 routing only one route between each source and destination pair is used. The O4 routing uses four different routes for each pair of source and destination nodes. A source node uses the four routes in round robin fashion for consecutive message packets sent to a destination. We simulated the adaptive routing with six different output selection functions: LRU, MRU, RND, RR, LRUC, and LRUD.

For a more comprehensive study, in addition to the random traffic, we used three other permutation traffic patterns: bit-complement, matrix-transpose, and bit-reversal permutations. In bit-reversal, a source processor represented in binary by $s_{n-1}s_{n-2}\dots s_1 s_0$ sends messages to destination $s_0 s_1 \dots s_{n-2}s_{n-1}$. For bit-complement, this source sends to $\overline{s_{n-1}s_{n-2}}\dots\overline{s_1 s_0}$. In matrix-transpose, the destination is $s_{\frac{n}{2}-1}s_{\frac{n}{2}-2}\dots s_1 s_0 s_{n-1}s_{n-2}\dots s_{\frac{n}{2}+1}s_{\frac{n}{2}}$.

We used two topologies, for 16 node and 128 node systems illustrated in Figs. 2-3 and six different message sizes, 128, 256, 512, 1K, 2K, and 4K bytes. We applied input loads from 0.1 to 1.0 in increments of 0.1. All combinations of these methods result in $6 \times 2 \times\ 6 \times 4 \times 10 = 2880$ individual simulation runs (each corresponding to a data point on the latency curves.) We ran each simulation for 1 million switch cycles. We chose a cold–start time of 100,000 cycles after which we started measuring latency. If measured average latency remains constant then the network is in steady state, if it keeps increasing then it means that the network is saturated.

We ran all simulations on a 32 node RS/6000 SP farm at the T. J. Watson Research Center. Each simulation run takes tens to 100s of minutes depending on the load and size of the network. A total of 2880 datapoints would take tens of days if they were to be executed on a single computer. Thus, the availability of 32 nodes shortened this time significantly and allowed exploration of many different selection functions.

5.2 Simulation Results

Figure 5 illustrates the average message latency on a 16-node system under the bit-reversal traffic pattern. With one-route oblivious routing method (O1), the 16 node network saturates very quickly. In general, the one-route oblivious either performs very well or very poorly depending on the permutation. Due to its dismal worst case performance and greater performance variations, it is not being used in SP networks, and we do not consider one-route oblivious further in this paper.

Figure 5 shows the performance advantage of adaptive routing over four-route oblivious. With four-route oblivious the network saturates earlier than with the adaptive routing. In the rest of this section we will focus on the effect of output selection function on the performance of adaptive routing. We also present the results for the four-route oblivious along adaptive routing algorithms.

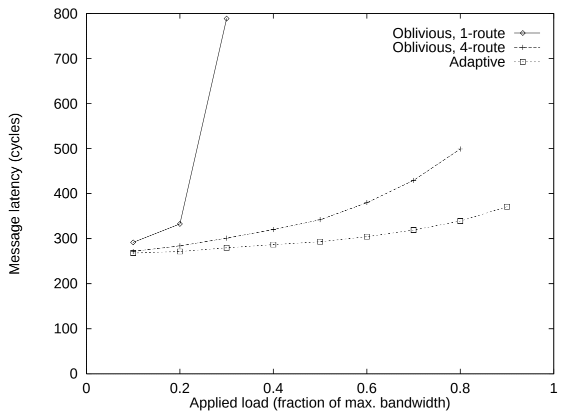

Figure 5: Adaptive routing and oblivious routing latency.

Figure 6(a) shows the peak throughput of the network (as a percentage of maximum possible) using the four-route oblivious routing and the adaptive routing algorithms with different selection functions and for 128-byte messages on a 16 node system under random traffic. All algorithms perform more or less the same for small message traffic. Figure 6(b) is for the same conditions, except message size is 4096-bytes, which shows that all adaptive routing algorithms perform the same regardless of the choice of output selection function. For long messages, LRUD outperforms the others negligibly. However, all adaptive algorithms perform better than the oblivious algorithm.

The results for the bit-reversal permutation shown in Fig. 7(b) show that the MRU selection function performs poorly for long messages. The throughput for the MRU selection function is 0.75 while the LRU, LRUC, and LRUD achieve a throughput of 0.99. The other selection function which performs slightly worse is the RND selection function. The peak throughput for 4096-byte messages for the RND selection function is 0.92.

Figures 8 and 9 illustrate the performance of the oblivious routing and the adaptive routing with different selection functions in a 128-node system. It can be observed that the adaptive routing algorithms outperform the oblivious routing. The performance of all of the selection functions are the same for 128-byte messages under the random traffic (Fig. 8(a)). The MRU and LRUC perform slightly worse than the other adaptive selection functions for long messages (Fig. 8(b)). The simulation results for the bit-reversal traffic is shown in Fig. 9. It can be observed that using the adaptive routing instead of the four-route oblivious routing results in more than 229% improvement in the peak throughput. It can be seen that for short messages LRU and LRUD outperform the rest of the selection functions. The RND, RR, and LRUC selection functions perform marginally worse than LRU and LRUD. The MRU selection function performs poorly. For longer messages (Fig. 9(b)), MRU and LRUC performs worse than the rest of the adaptive selection functions, achieving a throughput of 0.55 and 0.54, respectively. It can be seen that using LRU instead of LRUC increases the peak throughput by more than 30%.

To summarize the simulation results we can say that for the random traffic, all of the selection functions perform well and LRU, LRUD, and RR outperform MRU, RND, and LRUC only marginally. The performance of the studied selection functions show a wider range for the bit-reversal traffic. For this traffic pattern, LRU and LRUD outperform the other selection functions (LRUD performing slightly better than LRU). The RR selection function shows a good performance under the bit-reversal traffic as well. The MRU selection function performs poorly across the board, while LRUC and RND functions perform poorly for long messages. In the next section, we discuss these results in detail.

5.3 Discussion

In this section, we first discuss the performance of the studied selection functions and draw some conclusions. Then, we discuss the hardware complexity of implementing these output selection functions. We finally discuss which selection function seems to provide the best performance with a reasonable hardware complexity.

One of the major observations is that each of the studied selection function's performance is consistently same relative to others under varying conditions. Those selection functions which perform well and achieve the highest peak throughput consistently do so for different system sizes, message sizes and traffic patterns. This behavior is different from that of selection functions studied on the mesh torus networks [9, 13]. This characteristic may be attributed to the fact that in BMIN, unlike meshes and tori, most nodes are equidistant. Unlike meshes and tori, there is no "center" in BMIN where traffic needs to be routed around. BMIN performance is less sensitive to mapping of tasks to nodes since nodes are equidistant mostly. This leads us to the conclusion that only one output selection function can be used in all of SP systems. There is no need to use hybrid selection functions to improve the performance of the system.

The BMIN used in SP systems is essentially a fat tree interconnect. Messages sent from the source node traverse uplinks to reach a least common-ancestor switching element. There are multiple (e.g. 4) output ports that can be taken at each of the switching elements before reaching a least common-ancestor switching element. When the message arrives at a least common-ancestor switching element there is only one unique path (a set of downlinks) that can be taken towards the destination node. In most cases, messages can find a free output port while going up the tree by using adaptive rout-

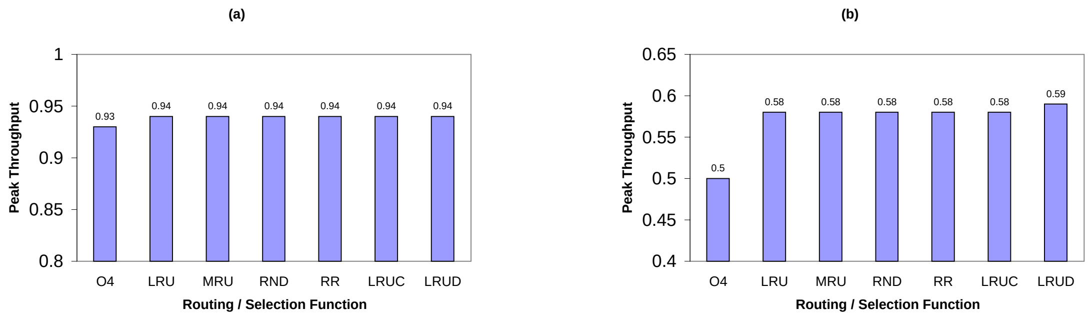

Figure 6: Peak throughput for the oblivious routing and adaptive routing with different selection functions on a 16-node system with random traffic pattern for (a) 128-byte messages and (b) 4096-byte messages.

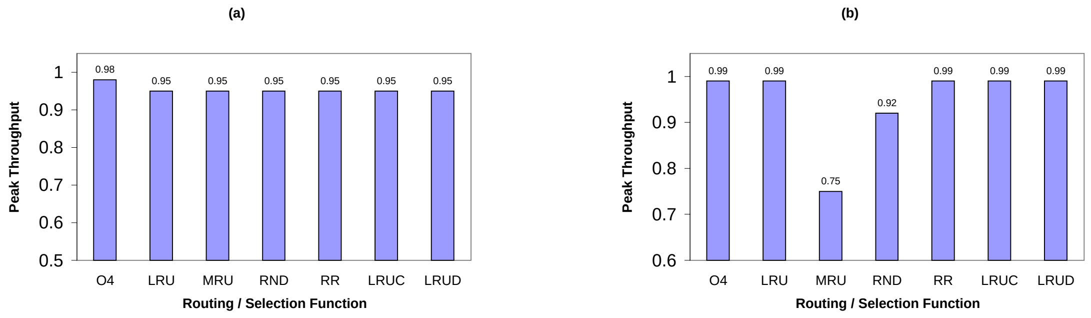

Figure 7: Peak throughput for the oblivious routing and adaptive routing with different selection functions on a 16-node system with bit-reversal traffic pattern for (a) 128-byte messages and (b) 4096-byte messages.

ing. However, when a message is traversing the downlinks, if the desired output port is busy the message doesn't have any other choice of output and has to wait. Therefore, it is crucial to direct messages along the uplinks such that the congestion and the chance of messages getting blocked while messages are using the downlinks is minimized. In the absence of any global knowledge about the traffic on different links of the interconnect, the best way to avoid congestion seems to be distributing the messages for any given source–destination pair to all of the least common-ancestor switching elements of the pair. In order to achieve this goal, choosing a selection function which distributes the traffic among the switching elements of the BMIN (or fat tree) evenly becomes important.

The simulation results presented in Section 5.2 shows that the RND and MRU selection functions do not perform well. This can be attributed to the fact that when using these selection functions, the traffic does not get distributed evenly. MRU wants to repeatedly forward messages incoming from a given input port to one output port. RND, due to its random nature, forwards at random times two or more consecutive packets (i.e. multi-packet message) incoming from the same input port to one output port, thus not helping with the distribution of messages.

From the simulation results it can be observed that the LRUC selection function performs worse than LRU for long messages. As stated before, the LRU selection function uses an LRU list per input port and therefore consecutive packets incoming to the input port are forwarded to different output ports. On the other hand, the LRUC selection function uses a single centralized LRU list for all the input ports in the switch chip. This does not help with the distribution of messages to the output ports. To see this by example, consider any input port I_x that forwards a packet to output port O_y. Now the port O_y is the most recently used output, hence it becomes the last element in the central LRU list of the switch chip. However, at high workloads by the time another packet arrives at I_x, incoming messages from other inputs would have moved O_y up to the head of the LRU list hence making O_y the least recently used output. Therefore, I_x finds O_y as the LRU output again and sends the message to the same output port O_y again. This may result in hot spots downstream from O_y.

As mentioned earlier in this section, congestion in the SP network mostly happens in the downlinks. In particular, messages going toward the same switching element connected to a set of destination nodes will compete with each other for the same resources. In order to distribute this traffic among all links and switching elements and reduce the congestion, we used the LRUD selection function. In this method, the selection of output ports in the upward paths to least-common ancestor switches is based on the destination switching element. By using a separate LRU list for each destination switching element, we try to distribute the traffic evenly among all possible uplinks and therefore all least common-ancestor switching elements. From the simulation results it can be observed that LRUD outperforms LRU. However, the obtained improvement

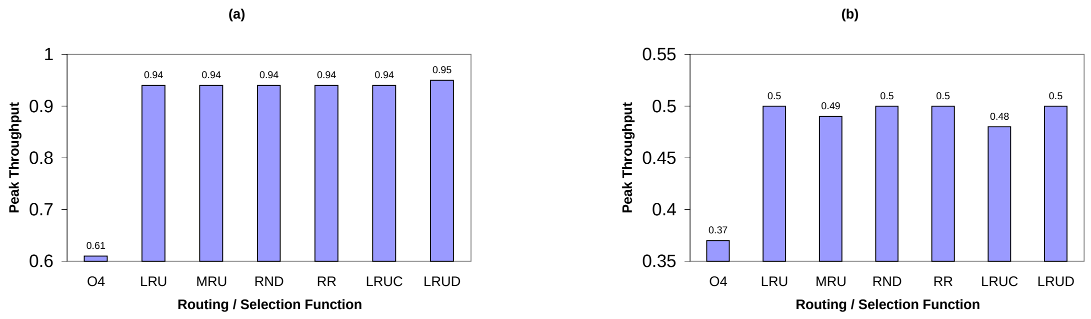

Figure 8: Peak throughput for the oblivious routing and adaptive routing with different selection function on a 128-node system with random traffic pattern for (a) 128-byte messages and (b) 4096-byte messages.

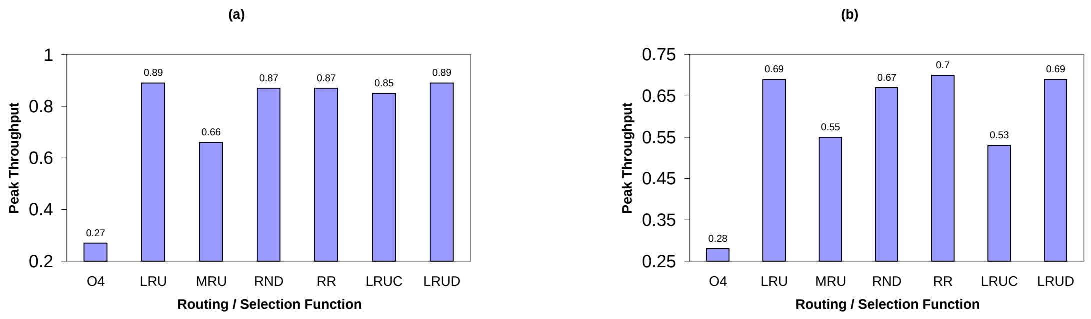

Figure 9: Peak throughput for the oblivious routing and adaptive routing with different selection function on a 128-node system with bit-reversal traffic pattern for (a) 128-byte messages and (b) 4096-byte messages.

is negligible. LRUD requires the switch chip to examine and determine the final destination from the packet header unlike source routing. It also requires lot more resources than other selection functions; in each switch chip for every destination switch in the network, one LRU list is needed. The marginal performance improvement of LRUD is not worth the extra hardware complexity.

From the simulation results it can be observed that the performance of the RR scheme is comparable to that of the LRU and LRUD schemes and in most cases is as good as them. Therefor, we can conclude that LRU, RR, and LRUD schemes performs better than other schemes.

Another important factor in choosing a selection function is the cost of the hardware implementation. Considering the number of the LRU lists required for implementing the LRUD selection function and the LRU selection function, it's clear that the LRU selection function is a better choice than LRUD. The implementation of the RR scheme requires less chip area than the LRU scheme. To be more specific, the implementation of the LRU scheme on IBM RS/6000 SP switch chips require $n(n+1)/2$ latches or flip-flops, $n(n-1)$ 2-input NAND gates, and n n-input NAND gates where n is the number of switch ports (e.g. n=8 for Switch2). A faithful implementation of the RR function requires a very similar amount of combinational logic, but only requires $\log_2 n$ latches. As latches are typically far more area-intensive than 2-input NAND gates, the area required for latches is a worthwhile consideration. For instance, in one of IBM's recent commercially available ASIC processes, scan-able latches required between two and five times the area of 2-input NAND gates, depending on the drive strength of each individual cell. In addition, reducing the number of latches can reduce chip testing time.

Based on the performance and hardware cost of the studied output selection functions, it can be concluded that the RR output selection function is the best one among these output selection functions.

6 Related Work

Previous studies on output selection functions have been only on the mesh and torus networks [3, 7, 9, 10, 13, 20, 21]. Schwiebert and Bell study routing algorithms named *opt-y*, *zigzag*, and *outside* for 2-D meshes [13]. Opt-y uses one virtual channel in the x-dimension and two virtual channels y1 and y2 in the y-dimension. Six different selection functions are obtained by taking permutations of x, y1, and y2. Authors conclude that selection functions have substantial impact on performance. Their results show that the zigzag selection function, which is theoretically optimal, does not perform well in practice since it concentrates traffic in the center of mesh for many communication patterns. Authors also show that the selection function which prioritizes output ports in the x,y1,y2 order has the best performance.

Feng and Shin study selection functions in 2-D square meshes and tori [9]. They simulate *dimension-ordered*, *random*, and *diagonal* selection functions for oblivious and adaptive routing algorithms. They use *random-uniform*, *bit-complement*, *bit-reverse*, and *transpose* traffic patterns as the workload in their simulations. They also study *deflection* routing algorithms which use non-minimal paths. These authors also conclude that selection functions have substantial impact on performance. For random-uniform and bit-complement traffic on the mesh, the results show that the dimension-ordered oblivious algorithm performs better than three different adaptive (minimum path) algorithms each with different selection functions. For transpose traffic on the mesh, their results show that the adaptive algorithm with random selection function performs better among all. For bit-reversal traffic on the mesh, their results show that the adaptive algorithm with diagonal selection function performs best among all. In conclusion, they suggest that the choice of selection function must be tailored to the network traffic on the mesh and torus. Our results show that this is not required on BMINs.

7 Conclusions

In this paper, we have presented the adaptive routing scheme used in the recently developed SP switch chip Switch2. We have shown that adaptive routing schemes outperform the oblivious routing methods. It is shown that adaptive routing increases the throughput by up to 229% over oblivious routing in some cases. We have also presented a comprehensive study of output selection functions and their impact on the system performance. We have shown that the LRU, RR, and LRUD output selection functions outperform the other selection functions consistently. Unlike the mesh and torus networks, in BMIN networks there is no need to use multiple (or hybrid) selection functions to obtain the best performance. We have also provided an analysis of the cost-effectiveness of different selection functions with respect to the complexity of their hardware implementations. We have concluded that the RR selection function is the most cost-effective selection function while, the LRU and LRUD selection functions provide a slightly better performance with increased hardware cost. Currently, we are studying output selection functions in BMINs which take into account waiting time of messages blocked in output ports.

References

[1] B. Abali and C. Aykanat. Routing Algorithms for IBM SP1. *Lecture Notes in Computer Science, Springer-Verlag*, 853:161–175, 1994.

[2] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali. Adaptive source routing in multistage interconnection networks. In *Proc. 10th Int. Parallel Processing Symp. (IPPS'96)*, pages 258–267, April 1996.

[3] S. Badr and P. Podar. An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh Connected Topologies. *IEEE Transactions on Computers*, 38(10):1362–1371, 1989.

[4] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware-Software Approach.* Morgan Kaufmann, March 1998.

[5] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, March 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, May 1987.

[7] J. Duato. Improving the Efficiency of Virtual Channels with Time Dependent Selection Functions. *Parallel Arch. and Lang. Europe 92*, pages 635–650, 1992.

[8] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach.* The IEEE Computer Society Press, 1997.

[9] W. Feng and K. G. Shin. Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks. In *Proc. 11th Int. Conf. Supercomputing*, 1997.

[10] C.J. Glass and L. M. Ni. The turn model for adaptive routing. *J. Assoc. Comp. Machinery*, 41(5):874–902, 1994.

[11] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. on Computers*, C-34(10):892–901, Oct. 1985.

[12] I. D. Scherson and C.-H. Chien. Least Common Ancestor Networks. In *Proc. 7th Int. Parallel Processing Symp.*, pages 507–513, 1993.

[13] L. Schwiebert and R. Bell. The Impact of Output Selection Function Choice on the Performance of Adaptive Wormhole Routing. In *Proc. 10th Int. Conf. Parallel and Distributed Computing Sys.*, pages 539–544, 1997.

[14] L. Schwiebert and D. N. Jayashima. Optimal Fully Adaptive Minimal Wormhole Routing for Meshes. *J. Parallel Distrib. Comput.*, 27(1):56–70, May 1995.

[15] M. Snir, P. Hochschild, D. D. Frye, and K. J. Gildea. The communication software and parallel environment of the IBM SP2. *IBM Systems Journal*, 34(2):205–221, 1995.

[16] C. B. Stunkel, J. Herring, B. Abali, and R. Sivaram. A New Switch Chip for IBM RS/6000 SP Systems. In *Supercomputing 99 (SC99)*, 1999.

[17] C. B. Stunkel, D. G. Shea, B. Abali, M. M. Denneau, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, M. Tsao, and P. R. Varker. Architecture and implementation of Vulcan. In *Proc. 8th Int. Parallel Processing Symp.*, pages 268–274, April 1994.

[18] C. B. Stunkel, D. G. Shea, D. G. Grice, P. H. Hochschild, and M. Tsao. The SP1 High-Performance Switch. In *Proc. 1994 Scalable High-Performance Computing Conference*, pages 150–157, May 1994.

[19] C. B. Stunkel*et al.* The SP2 High-Performance Switch. *IBM Systems Journal*, 34(2):185–204, 1995.

[20] J. Updahyay, V. Varavithya, and P. Mohapatra. A Traffic Balanced Adaptive Wormhole-Routing Scheme for Two Dimensional Meshes. *IEEE Transactions on Computers*, 46(2):190–197, 1997.

[21] J. Wu. An Optimal Routing Policy for Mesh-Connected Topologies. In *Int. Conf. Parallel Proc.*, pages 267–270, 1996.