RC 19914 (88122) 1/23/95
Computer Sciences/Mathematics 25 pages

IBM Research Report

The SP2 Communication Subsystem

Craig B. Stunkel, Dennis G. Shea and Bülent Abali

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598

Mark G. Atkins, Carl A. Bender, and Don G. Grice

IBM Highly Parallel Supercomputing Systems Laboratory
Neighborhood Road
Kingston, NY 12401

Peter H. Hochschild, Douglas J. Joseph, Ben J. Nathanson, and Richard A. Swetz

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598

Robert F. Stucke

IBM Highly Parallel Supercomputing Systems Laboratory
Neighborhood Road
Kingston, NY 12401

Michael Tsao and Philip R. Varker

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

This page intentionally left blank for double-sided printing

an edited version appears in the IBM Systems J., vol. 34, no. 2, pp. 185-204, 1995.

The SP2 High-Performance Switch

Craig B. Stunkel, Dennis G. Shea, Bülent Abali
Mark G. Atkins†, Carl A. Bender†, Don G. Grice†
Peter H. Hochschild, Douglas J. Joseph, Ben J. Nathanson
Richard A. Swetz, Robert F. Stucke†, Michael Tsao
Philip R. Varker

IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

†IBM Highly Parallel Supercomputing Systems Laboratory
Neighborhood Road
Kingston, NY 12401

Feb. 21, 1995

Abstract

The heart of an IBM SP2[1] system is the High-Performance Switch, which is a low latency, high bandwidth switching network that binds together the system's RISC System/6000[1] processors. The Switch incorporates a unique combination of topology and architectural features to scale aggregate bandwidth, enhance reliability, and simplify cabling. It is a bidirectional multistage interconnect driven by a common oscillator, and delivers both data and service packets over the same links. Switching elements contain a dynamically-allocated shared buffer for storing blocked packet flits. The SP2 communication adapter uses a variety of techniques to improve bandwidth and offload communication tasks from the node processor. This paper examines the SP2 Switch architecture and overviews its support software.

[1]A trademark or registered trademark of International Business Machines

This page intentionally left blank for double-sided printing

1 Introduction

The heart of an IBM SP2 system is the High-Performance Switch, which is a low latency, high bandwidth switching network that binds together the system's RISC System/6000 processors. The Switch is designed to provide near-constant bandwidth per processor for a large range of system sizes. The SP2 is available in configurations from 4 to 128 processors, with systems up to 512 processors currently available by special request.

The dominant goals for the SP2 communication subsystem are scalability, modularity, and ease of integration with the processing nodes. The objective for scalability is a network that linearly increases its aggregate bandwidth as the number of nodes[2] increases, while maintaining low average latency for message transmission. Fault tolerance is an integral element of scalability, because the large potential size of these networks insures that faults and errors will occur. The goal for modularity is to provide cost-effective networks for small systems that function as building blocks for larger systems. Finally, we required the ability to quickly integrate (attach to) the latest processor technology. Considerations here include achievable user-to-user bandwidth and latency and the complexity of communication protocol design. The SP2 communication adapter—the interface between an RISC System/6000 and the switching elements of the network—provides several features that optimize the bandwidth of the processor-to-network connection and reduce the message processing overhead for the processor.

The choice of *topology*—the pattern in which network devices are connected to provide communication—is one factor in achieving both scalability and modularity. There are a variety of topologies that have been chosen for connecting existing commercial parallel systems, as well as hundreds proposed in the literature.

At one extreme, bus-based systems are well suited to connecting small numbers of nodes, but are limited by a total bus bandwidth that does not increase as more processors are added. Bus-based solutions also suffer from the electrical disadvantages of more signal drops and longer transmission paths as processors are added. They are therefore not appropriate for connecting hundreds of nodes.

To combat bus scalability problems, massively parallel processors (MPP's) use point-to-point interconnection networks: their networks are constructed by connecting *switching elements* by point-to-point links (where each link connects only two devices). Herein, a switching element is defined to be a device with multiple input and output ports that forwards packets[3] arriving at an input port to a desired output port. Furthermore, switching elements are assumed to be *non-blocking*; that is, if a packet arriving at an input port x is destined for a particular output port y, and no other received packets are destined for y, then this packet may be immediately forwarded to y, regardless of other received packets.

[2]Herein, nodes are defined as processors or I/O servers which are the source or destination of messages over the High-Performance Switch. For SP2 systems, all nodes contain a RISC System/6000 processor.

[3]Messages are typically broken into smaller units called packets. In the SP2 Switch, each packet is self-routing: besides containing message data, it contains sufficient information for the network to route the packet to its intended destination.

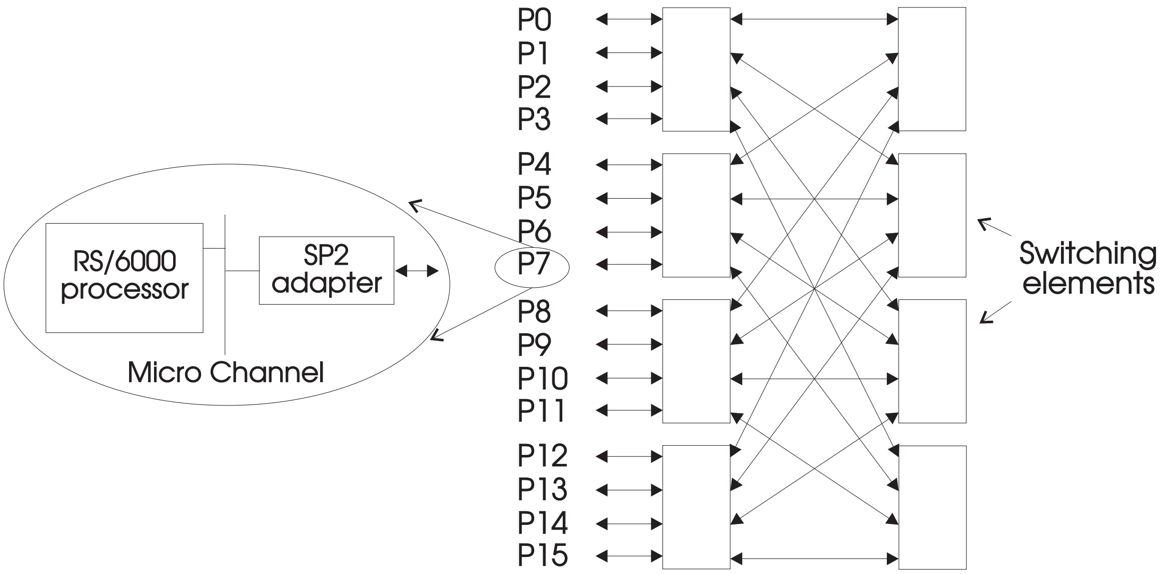

Figure 1: A 16-node SP system

A particularly advantageous network would connect every node to a single large switching element, and would thus provide non-blocking communication between all pairs of nodes. However, given n nodes, switching elements cannot be constructed to connect large n because their internal complexity increases proportionally to n^2, and their number of chip I/O increases proportionally to n.

Therefore, in MPP's, scalable bandwidth is achieved by interconnecting multiple small switching elements via point-to-point links [1, 2]. In the SP2 Switch, these links are bidirectional, comprising two channels for communicating data in both directions simultaneously. The aggregate bandwidth of an MPP is typically scaled by increasing the number of switching devices in the network, in a manner dependent upon the chosen topology.

An SP2 system, an example of which is shown in Figure 1, employs a unique combination of features to attain the SP2 goals for scalability, modularity, and processor integration. Its topology is a bidirectional multistage interconnection network, which scales bandwidth linearly while providing redundancy that increases with system size. The network is driven by a common oscillator, removing clock boundaries inside the network and resulting in improved reliability. This common clock is also the key to providing a common time-of-day across the entire network, visible to the user level at each processor. Service and diagnostic operations utilize the same links as data traffic, leveraging the power and redundancy of the Switch network and reducing the number of components.

The switching element of the network—the Vulcan switch chip—contains a large shared dynamically-allocated buffer for storing blocked packets. The switch chip implements a powerful flow control technique we call buffered wormhole routing to minimize latency and maximize throughput. Packets that encounter no contention inside the switch chip traverse the chip in a few cycles, while those that are blocked are temporarily stored in the central buffer.

The peripheral component of the network, the SP2 communication adapter, is the interface between a node's RISC System/6000 processor and a switching element. The SP2 communication adapter uses a variety of techniques to improve bandwidth and offload communication tasks from the node processor. It contains its own microprocessor, providing flexibility in protocol implementation.

In this article we examine the architecture and implementation of the SP2 communication subsystem, beginning with the basic elements of the network and then building up to large topologies. The properties of an SP2 channel are described in §2. In §3 we profile the Vulcan switch chip, followed by a discussion of the SP2 communication adapter in §4. We finish our network description by examining the chosen family of SP topologies in §5, and §6 provides simulation results that support our claims of scalable network bandwidth. An overview of the Switch support software is given in §7.

2 SP2 Switch channel properties

Each SP2 link contains two channels carrying packets in opposite directions between two network devices. In this section we will examine the communication protocol and other properties of an SP2 Switch channel. First we present an overview of the SP2 packet flow control strategy (the method by which packets are blocked or allowed to proceed without loss of packet data). We then describe the network's common oscillator, which is crucial to the understanding of the channel protocol, followed by a brief explanation of the individual channel signals.

Packet flow control

SP2 packet flow control is related to *wormhole routing* [3]. In wormhole routing, flow control is performed on units that are smaller than packets: flow-control digits, or *flits*. The head (first flit) of the packet advances immediately through each switching element unless it is blocked by contention for an output port, and succeeding flits of the packet advance in pipelined fashion behind the head. This immediate forwarding minimizes the latency per switching element. When the packet head is blocked, all flits of the packet are buffered in place until the output port is free. Thus, a single blocked packet may be blocked in place across many switching elements.

SP2 flow control improves on this scheme by allowing succeeding packet flits (for SP2 a flit is one byte) to advance as far as the switching element that contains the blocked packet head. The Vulcan switch chip's central queue is the primary mechanism for this advancement, allocating as much storage as possible for the succeeding flits of the blocked packet. Thus, often the entire blocked packet can be stored within a single switching element, freeing links that would have remained allocated and idle for wormhole routing. There is no requirement, however, that a switching element that accepts a packet header be able to buffer the entire packet, as required in virtual cut-through flow control [4]. To emphasize these advantages over ordinary wormhole routing, we call the SP2 flow control *buffered wormhole routing.*

Common network oscillator

All SP2 Switch hardware is synchronous, and the entire network is clocked from a common 40 MHz oscillator. At the hardware level, this improves the reliability of the Switch hardware by eliminating clock boundaries. But the results are visible to both operating system and user as well. Each processor node (on its SP2 adapter) maintains a local time-of-day register in synchronism with other nodes, without concern that its time-of-day will drift to respect to the other nodes.

This SP2 global time provides a simple and extremely accurate solution to the well-known distributed clock synchronization problem [5]: how to align the separate time-of-day clocks as closely as possible throughout the system. Global time also offers advantages in fault detection, performance monitoring, and debugging. Link error checking is performed on a time basis, rather than a per-packet basis, permitting errors to be detected in bounded time. A global event trace of a parallel application can be synthesized from local traces maintained at each node. Timestamps provide for proper sequencing of database transactions.

It would be awkward for a large system if synchronous operation meant that clock and data cables had to be cut to restricted lengths in order to preserve timing relationships. In fact, clock and data cables on SP machines can have arbitrary lengths[4], through a combined hardware and software mechanism we call *tuning*. With tuning it is not necessary to adjust the phase of the common oscillator signal received by each device. Instead, each device adjusts the phase of data, using adjustable delay elements. The tuning mechanism [6], which measures cable length electrically, also facilitates the accurate synchronization of the nodes' time-of-day clocks [7]. In SP2, this mechanism is software-controlled and is contained in the Worm software package discussed in §7.

To avoid the common oscillator becoming a single point of failure for the SP2 Switch, there are multiple available oscillators in the network, and there is redundancy in the repowering network that delivers this clock to all network devices. One of the multiple oscillators is selected as the master, and is driven to the entire network via extra signals within the SP2 link cables.

[4]The longest cable length is set by electrical considerations and by a protocol maximum of 15 bits in transit on a cable at one time.

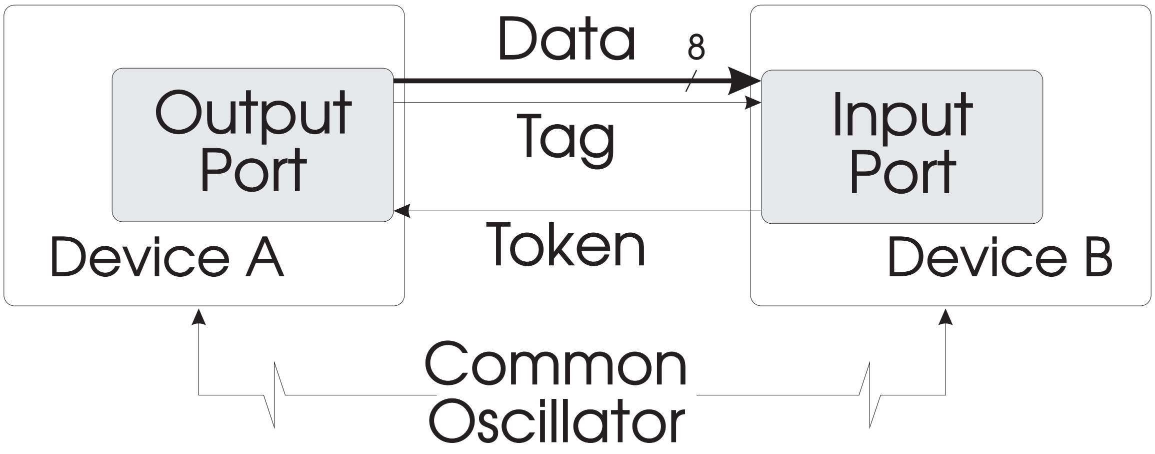

Figure 2: Switch ports connected by a channel

Channel protocol

In the SP2 Switch, packet transmission and flow control is accomplished via tag and token signals, shown in Figure 2. An active tag indicates a valid packet flit formed by 8 bits of data. Each output port begins operation with 31 tokens, which correspond to the 31 available flit-sized buffer spaces in its downstream input port's receiving FIFO. For each packet flit transmitted by the output port, the output port decrements its token count. Each cycle in which the input port transfers a flit out of its input FIFO, the input port sends a token back to the output port via the token signal. As long as the output port's token count is non-zero, the port can transmit packet flits.

This token methodology permits the use of long links without loss of link bandwidth—no immediate acknowledgement is required for each flit, and thus flits and tokens are traversing the link in opposite directions simultaneously. For long links, data and tokens are pipelined on the link—new data and token signals are placed on the link every cycle. To maintain peak bandwidth for a link with a p cycle propagation delay, the input port must contain a buffer of at least size $2p$. This buffer size handles the worst-case scenario in which p flits and p tokens are in transit simultaneously. With the Vulcan switch chip's 31-flit input port buffers, this corresponds to a limit of $p <= 15$ for SP2 links (although electrical concerns may place more restrictive limits on the maximum link length).

Network operation is divided into 64-cycle frames via the common time-of-day at each device, and 2 cycles out of each frame are used to send error detection codes. This time-based error detection mechanism permits timely error detection and accurate fault isolation even for dropped packets (e.g. from unplugged or severed cables or from loss of power in a network device). In addition, node software may embed additional error detection

Figure 3: The SP2 Switch packet

information within each packet, to be checked upon arrival at the destination node.

Packets, shown in Figure 3, vary in length up to 255 flits. The first flit holds the total packet length, in flits. The next flit or flits contain route information. The rest of the packet is message data. The length and route fields of the packet are often referred to as the packet header. Long messages are split into multiple packets that are $\leq$ 255 flits for transmission.

3 The Vulcan switch chip

The basic switching element of the SP2 High-Performance Switch is the Vulcan switch chip, shown in Figure 4. This chip contains 8 input ports and 8 output ports, each of which is one byte (one flit) wide, and each port can process one flit per cycle. The chip is clocked at 40 MHz, and therefore can receive up to 320 MB/s of packet data, and can transmit a maximum of 320 MB/s. In between the input and output ports are a crossbar for transferring packets that encounter no contention for their desired output port, and a large (1 KB) dynamically shared buffer called the *central queue* for storing the flits of packets that cannot proceed because of contention. This section will describe these Vulcan switch chip features in detail.

To describe the traversal of a packet through a switch chip, we again refer to Figure 3. When the header of a packet arrives at an input port, the first flit of the route field is inspected to determine the desired output port; a request is then made to this output port for forwarding through the chip crossbar. Each route flit embeds two 3-bit route fields and a selector bit that indicates which of the route fields holds the current route. Each route field is the 3-bit binary encoding of the desired output port, 0-7. After the second route field has been used, the route flit is thrown away and the packet length field is decremented. Hence, no route flits remain in the packet when it reaches its final destination.

If the output port grants the request from the input port, the packet flits are immediately forwarded through the flit-wide crossbar to the output port. This is the low latency path through the switch chip, and typically requires only 5 cycles (125 ns) from the time the first packet flit is received at the input port until this flit is transmitted out of the output port. An output port is granted to a packet for the duration of the packet.

If the output port request is not granted, the packet flits are temporarily stored in the central queue. Requests from two or more input ports for the same idle output port are handled on a least-recently-served basis. A request will also be denied if an output port is currently allocated by a packet from another input port, or allocated by a packet that has

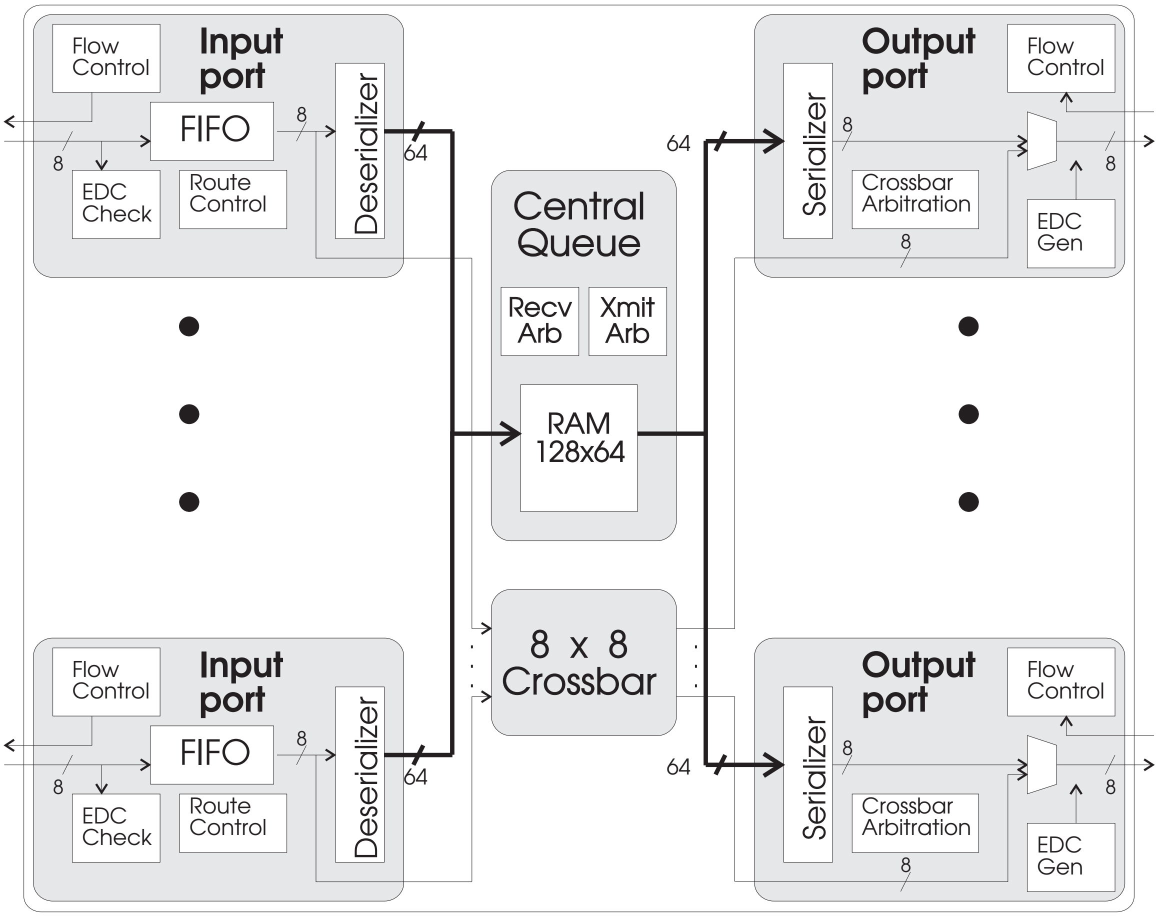

Figure 4: Vulcan switch chip organization

been stored within the central queue (i.e., packets within the central queue have priority over other packets).

The central queue actually maintains 8 queues—each of these queues is a dynamically-allocated FIFO (first-in first-out) queue, one for each output port. By dynamic, we indicate that the central queue contains one shared pool of buffer space which is allocated to input ports as requested. Busier input ports can use a larger portion of the central queue than lightly loaded input ports. Similarly, if packets from several input ports are destined for the same output port, then the queue assigned to that output port might consume a large portion of the total central queue buffer. This dynamic behavior typically improves network performance by allocating more resources to busier ports in the network.

This central queue organization effectively results in output port buffering, because

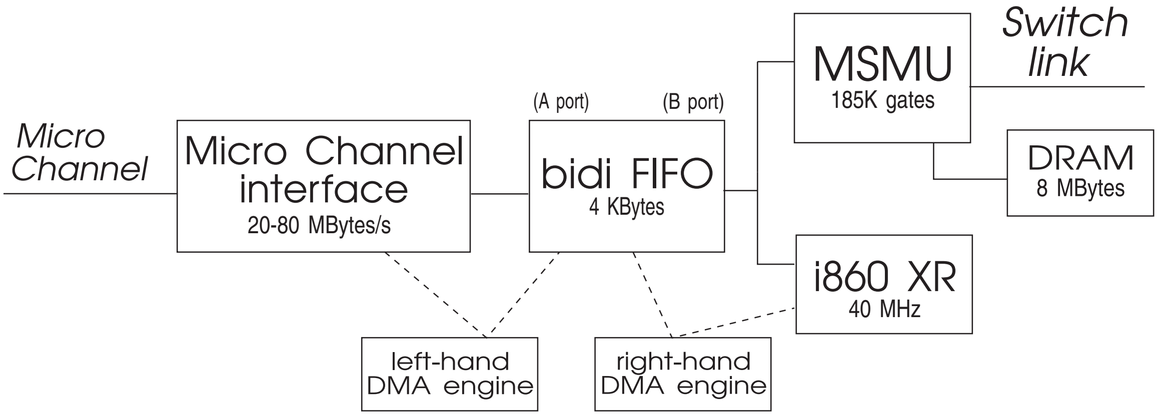

Figure 5: SP2 communication adapter

packets are effectively queued at the output ports when blocked. This type of buffering is known to be superior [8] to the more prevalent input port buffering. Input port buffering suffers from head-of-the-line blocking, in which a blocked packet in the input port FIFO stalls the progress of succeeding packets which may be destined for idle output ports.

The central queue is implemented as a dual port memory—it can perform one write and one read per cycle. In the worst case, each input port can be forwarding packet data to the central queue simultaneously. To match the maximum possible bandwidth from the input ports, it is necessary to write 8 flits per cycle into the central queue. Thus, each input port queues a *chunk* of 8 flits (a process called deserialization) before requesting service from the central queue, and writes the entire chunk in one cycle when the request is granted. Conversely, a serialization process occurs at the output port to convert an 8-flit central queue chunk read into the flit-wide data stream that is sent from the output port. Simultaneous requests from input ports (or output ports) are arbitrated by the central queue on a least-recently-served basis.

As long as the central queue is not full, each input port can continue to receive flits at full bandwidth. When the central queue fills, input ports with flits destined for the central queue will not be able to empty their input FIFO's, and eventually a lack of tokens will cause the associated upstream output port to block.

4 The SP2 processor communication adapter

In SP2 systems, processor nodes are attached to the switching elements of the network via communication adapters. The SP2 communication adapter, illustrated in Figure 5, offloads communication tasks from the node processor. Its primary function is to move data between the node memory and the switch. It incorporates an Intel i860 microprocessor, with 8 MB of four-way interleaved DRAM, for communications coprocessing. The SP2 adapter attaches to the Micro Channel, and uses the Micro Channel's bus master and streaming capabilities for a 80 MB/s peak rate. The Micro Channel, a standard bus used for connecting peripherals to IBM workstations and personal computers, provided an interface known at SP2 design time, and guaranteed to remain stable over a variety of RISC System/6000 releases. Faster communication could be achieved by interfacing the SP2 communication adapter to an internal machine bus, at the expense of longer design time and incompatibility with future RISC System/6000 machines. This decision was also a major contributor to the short time interval between the introduction of the first POWER2 workstation and the availability of the first SP2 system.

The SP2 adapter connects the i860 bus to one output port and one input port of the Switch network via an ASIC chip called the MSMU (Memory & Switch Management Unit) [9]. The MSMU contains a 2K byte FIFO for each port. These FIFO's are memory-mapped, as are various other registers that provide control for and status about transmitted and received packets. A set of programmable interrupts is provided for events such as packet reception, FIFO space threshold recognition, and error detection. As the MSMU acronym implies, this chip also serves as the memory controller for the i860 bus, generating and checking ECC for the memory words and performing DRAM refreshes.

Data transfers involve two buses—the Micro Channel and the i860 bus. The SP2 adapter joins these buses with a 64-bit-wide, 4KB bidirectional FIFO (the "bidi FIFO," 2KB to and from the Micro Channel) Each interbus data transfer then has two components: Micro Channel to/from FIFO, and i860 bus to/from FIFO. Independent state machines manage the two parts of the transfer—the "left-hand DMA engine" for the Micro Channel and FIFO; the "right-hand DMA engine" for the i860 bus. The adapter also provides a bidi FIFO bypass that allows the RISC System/6000 processor to address the i860 bus directly using PIO (programmed I/O) instructions, giving the main processor direct access to adapter memory. Address protection on the adapter protects sensitive global data structures from modification by users.

The i860 initiates all DMA transfers, a process that begins when the i860 writes a header to the FIFO. A header describes an operation to be executed by the DMA engine. On the Micro Channel side, the left-hand engine extracts the header when it reaches the head of the FIFO and performs the requested operation. The right-hand engine monitors the header writes on the i860 bus and performs right-hand DMA as required.

For receiving data from network the i860 writes the header, and then the right-hand DMA engine takes control of the i860 bus and transfers network data into the bidi FIFO. Hardware ensures that the i860 is taken off its bus immediately after writing the FIFO and remains off the bus until right-hand DMA completes. When the header reaches the head of the bidi FIFO, the left-hand engine decodes it and transfers the data in the FIFO onto the Micro Channel.

For sending data to the network the i860 writes a header requesting data from the Micro

Channel. When the header reaches the head of the bidi FIFO, the left-hand engine initiates a transfer into the FIFO and indicates completion by incrementing a count in hardware. In response, the i860 writes a header, which initiates a right-hand DMA transfer from the FIFO to the MSMU. Since only the i860 initiates DMAs, and DMA requests are performed in the order that they are issued, the i860 always knows which completed transfers are waiting at the FIFO's i860 port and how long each transfer is.

The SP2 adapter incorporates a CRC generator/checker on the i860 port of the FIFO, and this port is protected by parity.

Unidirectional bandwidth through the SP2 adapter is approximately 35 MB/s for POWER2 nodes. This bandwidth is equivalent to the Switch link capacity, considering packet overheads such as error detection codes, sequencing, and route flits. The bidirectional bandwidth (simultaneously sending and receiving) for the current message-passing software ranges up to 48 MB/s, constrained mostly by Micro Channel degradation for short transfers. In addition to increasing bandwidth, offloading data transfer frees the RISC System/6000 to perform computation. On the fastest available SP2 processor nodes, about 25 percent of the processor remains available during bidirectional communication, and about 40 percent during sends, though not all applications can exploit this.

Communication protocol and the SP2 communication adapter

Communication protocols are important because of their impact on message bandwidth and latency. The SP2 communication adapter presents many interesting tradeoffs to the communication protocol designer, and we shall highlight some of these tradeoffs in this section. Protocol designers are granted flexibility in investigating different protocol strategies via the on-board i860 processor. The 8 MB of on-board memory accommodates protocols which require large amounts of local message buffering.

The primary goal of the communication protocol is to provide reliable low latency and high bandwidth data transmission, as elaborated in the following list:

- Provide the lowest possible latency from sending application to receiving application for a single user. This implies user access to the adapter, since kernel calls add too much overhead.

- Minimize the RISC System/6000 overhead for sending and receiving messages. This is especially important for larger messages, where data transmission can be overlapped with processing.

- Provide a guaranteed message delivery for the user which implies recovery from network failures, and detection of node outages.

- Ensure that unbalanced or misbehaving applications do not block other message traffic both from the same node and from other nodes.

- Provide a flexible mechanism for flow control for applications where flow control is an issue. This includes supporting unexpected traffic such as seen in client/server models of computing as well as unbalanced and unpredictable communication patterns such as seen in simulations.

Maximizing bandwidths for the network and Micro Channel requires careful orchestration of DMA with protocol on the i860 and the RISC System/6000. One extreme is to only use the i860 to initiate DMA between the network and RISC System/6000 storage. The other extreme is to code the i860 to handle the message passing calls directly. An intermediate position is to code the i860 to present a set of *virtual channels* to the protocol layer on the main processor. Since the i860 has a slower clock cycle, much smaller cache, and greater cycles per instruction, some protocol jobs might be done faster using only the main processor, avoiding synchronization overhead. However, most protocols do not require large amounts of code, and the code tends to be very control intensive (conditionals are frequently encountered). This makes the i860's smaller caches less of a disadvantage, and reduces the advantage of the RISC System/6000's branch prediction, bringing relative processor power closer than might be expected.

The Switch channel and Micro Channel are roughly equal in raw sustainable bandwidth (both about 77 MB/s), and the i860 bus bandwidth is approximately double that (160 MB/s). Thus about half the i860 bus bandwidth must be used for DMA transfers, leaving the i860 with the remaining 80 MB/s to query MSMU and DMA status, load and store packet headers in the MSMU, initiate DMA, and handle cache line fetches and storebacks. It is therefore extremely beneficial to organize the i860 code so that the protocol "fits" in the i860's instruction cache during right-hand DMA. Four operations can be active on the adapter at the same time:

- Right-hand DMA
- Left-hand DMA
- MSMU switch access
- i860 protocol execution

The i860 must alternate between handling outbound and inbound traffic so as to keep both the MSMU and the bidi FIFO's from sitting full or empty for long periods, if there is communication pending. To do this it must switch between code sections at unpredictable times. When this occurs, a penalty must be paid in terms of register usage. That is, much of the i860 register stack must be saved and restored on each switch, even if no function call is involved.

To summarize, the SP2 communication adapter offloads communication tasks from the RISC System/6000 processor in a reliable manner. The adapter provides the capability to send and receive messages at the full network link bandwidth, which is approximately

35 MB/s when including the effects of error detection and other types of packet overhead. Current message-passing software has achieved a combined bandwidth of 48 MB/s for simultaneously sending and receiving messages. The message latency of the product-level SP2 message-passing software is about 39 μs, which includes both sending and receiving overhead [10].

5 SP2 system topology

The system network topology is crucial to the SP communication subsystem goals of low-latency, high-bandwidth, fault-tolerant communication. To support high-bandwidth communication, a desirable topology *scales* aggregate bandwidth linearly; that is, the system network delivers total communication traffic that is proportional to the number of nodes in the system. As nodes are added to such a system, the network bandwidth also grows proportionally.

A well-known network type, the *multistage interconnection network* [11, 12] (MIN), can provide scalable aggregate bandwidth using switching elements with a small, fixed number of ports. These networks add switching "stages" to increase aggregate bandwidth as the number of nodes increases. MIN's are particularly attractive because of their ability to linearly scale *bisection bandwidth*, a common and realistic indicator of aggregate network capacity. Bisection bandwidth is the maximum possible bandwidth across a minimum network bisection, where a bisection "cuts" the network into two parts containing equal numbers of nodes.

An SP2 network is a type of MIN. The structure of the network will be described starting with the basic switching elements, and proceeding to the larger network components derived from them. The SP networks are based upon the 8-input, 8-output Vulcan switch chips described in §3. A higher-level representation of these switching elements is shown in Figure 6(a). However, in an SP network, each point-to-point link is full-duplex bidirectional—each link comprises two channels which carry data in opposite directions simultaneously. Input and output ports of each switching element are thus paired together to connect to an SP link. For example, input port 0 and output port 0 are attached to the same link, to receive and transmit signals, respectively. Therefore, in subsequent topology figures, each link shown has this bidirectional attribute, and input and output ports of each switching element are paired together as in Figure 6(b). Figure 6(c) shows yet another bidirectional representation of the switching element. This representation, with 4 input-output port pairs on either side of the block, will prove useful for drawing SP networks. All three representations are functionally equivalent.

Some entry-level SP2 High-Performance Switch (HPS) networks contain exactly one switching element mounted on a small board. Up to 8 processors can be attached to each of the bidi ports of the board via discrete cables that serve as bidirectional links. SP machines with more than 8 processor nodes are always constructed using larger switch boards. These larger SP2 switch boards contain two fully interconnected columns, or stages, of 4 switching

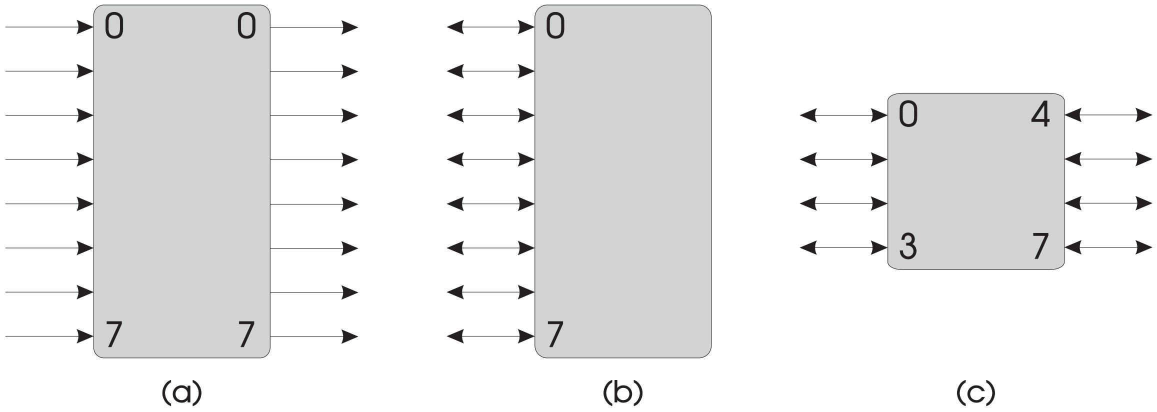

Figure 6: Alternate representations of an 8-input port, 8-output port switching element

elements, as shown in Figure 7. Note that there is a path between any two external links, providing connectivity between up to 32 externally connected devices. These boards are the building blocks used in constructing larger SP networks. For supporting large SP systems, these boards are advantageous over the smaller single-element boards because they incorporate more links internally and require fewer total boards, increasing network reliability and decreasing cost.

SP nodes are grouped into 16-processor units which are connected to one side of the switch boards (also called node switch boards because of their proximity to the nodes). Figure 8 displays a 16-processor SP system containing one node switch board. This figure also illustrates possible shortest-path routes for packets sent from processor 0 to two destinations. Note that processor 0 can communicate with processors 1, 2, and 3 by traversing a single switching element, and to the other 12 processors by traversing 3 switching elements.

The 16 unused links on the right side of the node switch board shown in Figure 8 are used for creating larger networks in one of two ways: (1) for systems containing up to 80 nodes, these links connect directly to the right sides of other node switch boards, and (2) for networks containing more than 80 processors, these links connect to additional stages of switch boards.

To illustrate strategy (1) for connecting systems of up to 80 nodes, Figure 9 shows the direct connection of the right-side links of 3 node switch boards to form a 48-way system. Example routes from node 0 to nodes 3, 10, and 35 are shown. Just as for a 16-way system, packets traverse 1 or 3 switching elements when the source and destination pair are attached to the same node switch board. When the pair are attached to different node switch boards, the shortest-path routes contain 4 "hops" (i.e., traverse 4 switching elements). For any pair of nodes connected to separate boards in a 48-way system, there are 8 potential paths, providing a high level of redundancy. One SP2 design goal is to provide at least 4 separate

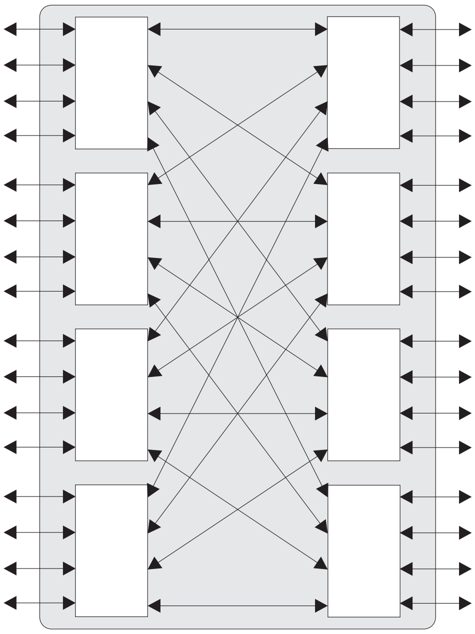

Figure 7: The SP2 switch board

paths from any source to any destination, except for nodes that are attached to the same switching element.

Given more than 80 nodes, this strategy of directly connecting node switch boards fails to provide the acceptable level of 4 redundant paths, and strategy (2) is instituted. Additional switch boards (termed intermediate switch boards) are cascaded to the node switch boards, effectively adding more stages to the network to scale the aggregate bandwidth. For example, a 128-way system is shown in Figure 10. This figure also displays routes from node 0 to node 31 and node 80. Destinations within the same 16-way group are reached in 1 or 3 hops as for smaller networks. Destinations not in this group, but on the same 64-way "side" of the network, are reached in 5 hops. It takes 6 hops to reach nodes on the opposite side. Note that every pair of stages in the network is connected by an equal number of links, preserving aggregate bandwidth. As the size of the SP2 network increases, so does its maximum redundancy. For example, there are 16 5-hop paths from node 0 to node 63.

These SP topologies are related to "least common ancestor" and fat-tree networks [13, 14, 15] in which each packet travels into the network only until reaching a switching element that is a least common ancestor of both the source and the destination. At this point it "turns" and travels back toward the destination. This implies an odd number of hops (because the turn, which occurs at the midpoint of the path, is done within a switching element). In contrast, for many of the SP networks (e.g., the 128-way) the midpoint of

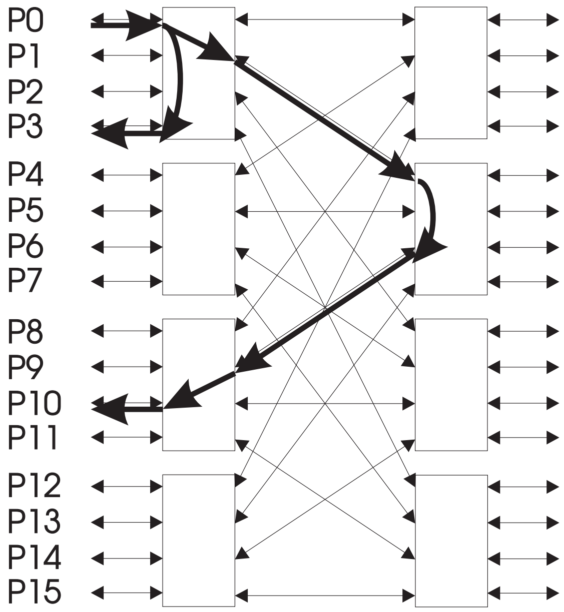

Figure 8: A 16-node SP system with example packet routes from node 0

the longest paths is a link instead of a switching element, resulting in an even number of hops. However, both SP networks and least common ancestor networks are provably deadlock-free for shortest-path routes, given that destination nodes guarantee to accept all packets presented to them by the network. This property allows packets to traverse any of the redundant shortest-path routes between pairs of nodes. Hence, the chance of creating extremely congested areas in the network ("hot spots") can be reduced by spreading packets over various redundant routes.

6 Switch performance

In §5 we made the claim that the SP2 Switch scales its aggregate bandwidth linearly as the number of nodes increases. This assertion is based primarily upon the choice of topology and on the design of the Vulcan switch chip. In this section we present simulation results to judge the scalability assertion by quantifying the effect of the central queue scheme, and by comparing SP topologies to a less scalable topology, the two-dimensional (2D) mesh.

We rely on simulation rather than system measurement because we are making performance comparisons to theoretical alternatives. In addition, we are interested in evaluating the performance of the network hardware separately from the complex effect of software message-passing protocols and their embedded end-to-end flow control mechanisms. Lastly,

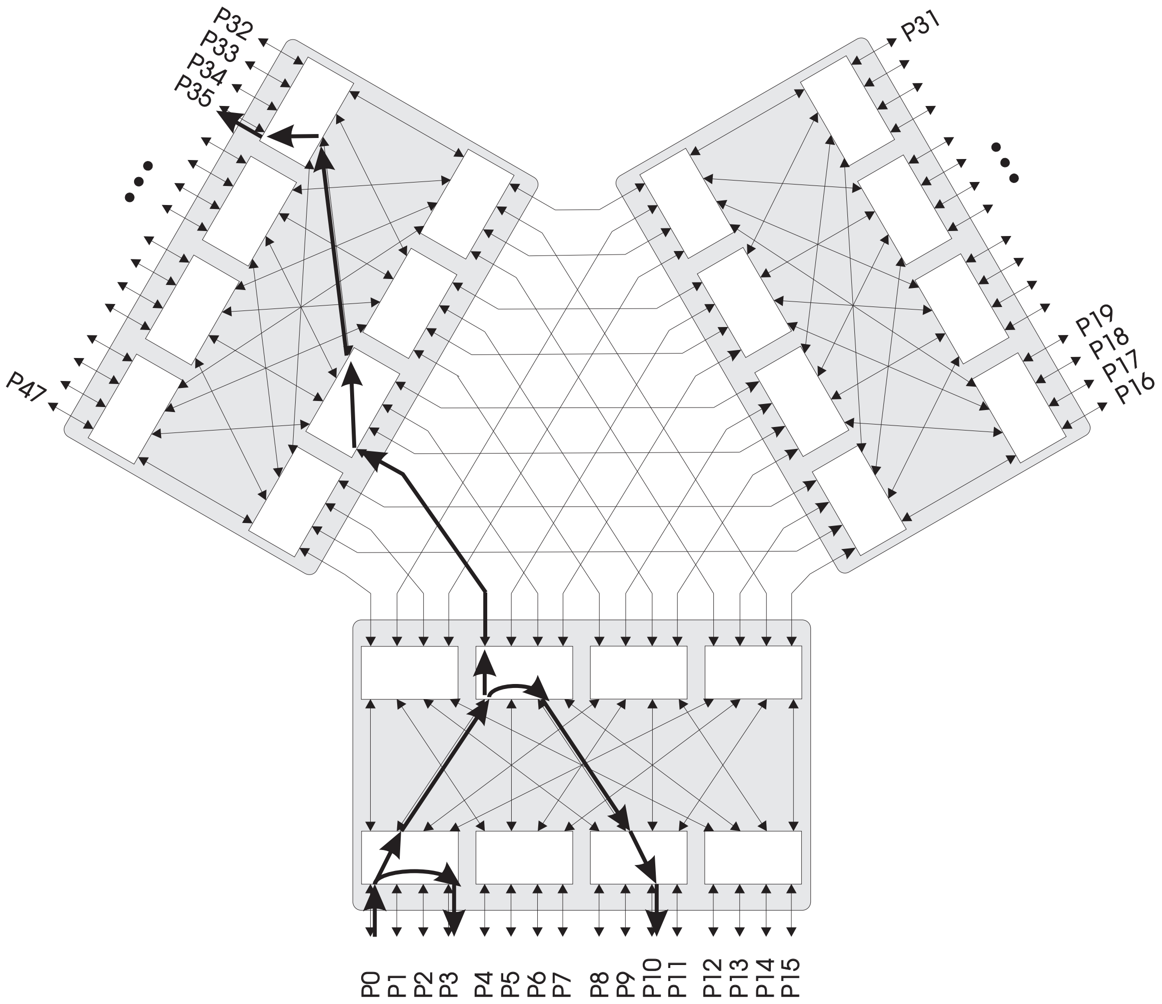

Figure 9: SP2 48-way system interconnection with example packet routes from node 0

we can stress the network to a greater degree in simulation than in the real system. However, system measurements presented within other papers in this special issue [16, 17, 18] give additional evidence of scalable performance.

The simulations are based upon a model of the Vulcan switch chip which closely mimics its register-level operation. Nodes are assumed to packetize messages—satisfying the 255-byte packet size limit of the Switch—and to transmit each packet into an infinite FIFO at the node's interface to the switching network. We assume that message flits are immediately pulled from the network upon arrival at a node. We simulated with random message traffic for varying message sizes. Destinations are chosen uniformly, with exponentially distributed

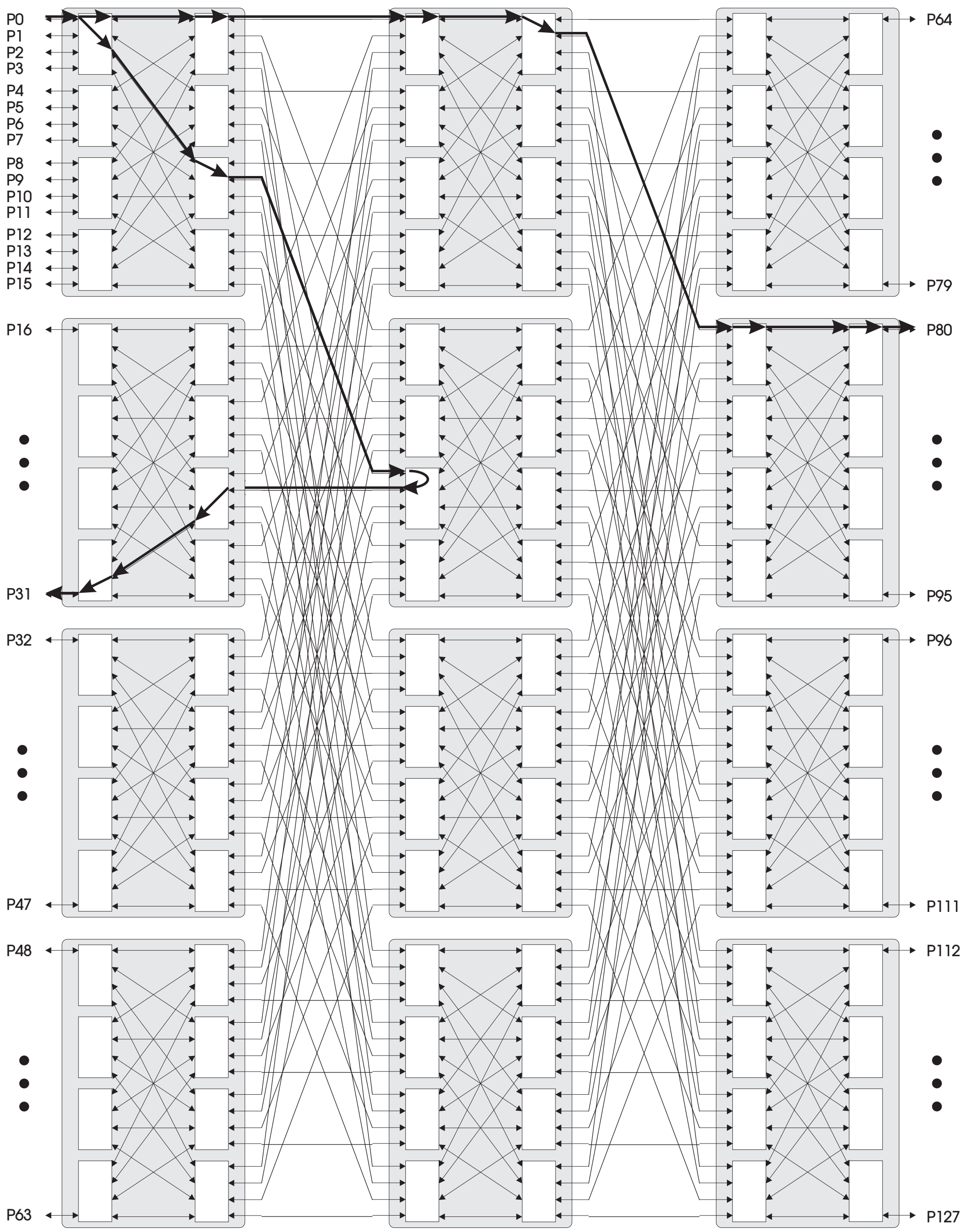

Figure 10: SP2 128-way system interconnection

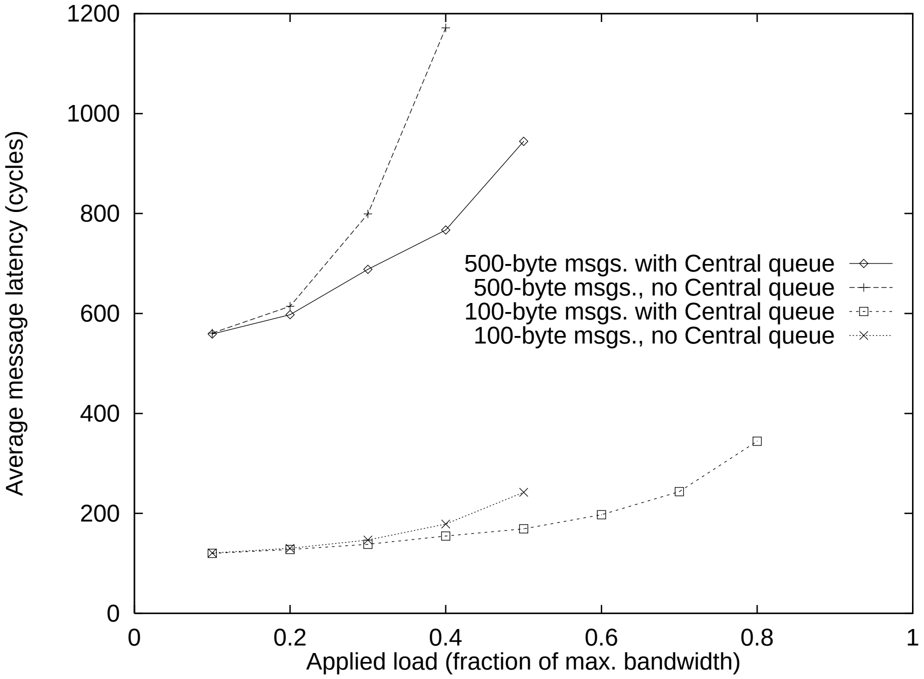

Figure 11: Average latency for central queue versus no central queue for a 16-way system

message transmission times. Latency results include queuing time at the source node.

First, we investigate the effect of the dynamically-allocated central queue. The Vulcan switch chip contains a 31-byte FIFO at each input port, and a 1 KB central queue, for a total of 1272 bytes of storage. We compare the Vulcan design to a switching element with no central queue, but containing 159-byte FIFO's at each input port, giving an equal amount of total buffer space. Figure 11 displays the average latencies encountered by messages with each type of switching element for a 16-node system. Simulations were conducted for input loads varying from 0.1 to 0.9 of the maximum bandwidth each node could drive into the network, and for two message sizes: 100 bytes and 500 bytes. Results are displayed only for simulations that did not saturate the network. The central queue simulations saturate at higher bandwidths and provide lower average latency for a given input load. The superior performance of the central queue is due to two factors: the "output buffering" effect referred to earlier, and the ability of the dynamic allocation scheme to present more buffer space to heavily used ports.

Lastly, we judge topological scalability by comparing the relative performance of a 16-way and a 128-way SP2 topology with a 16-way and a 49-way 2D mesh. We assume minimal

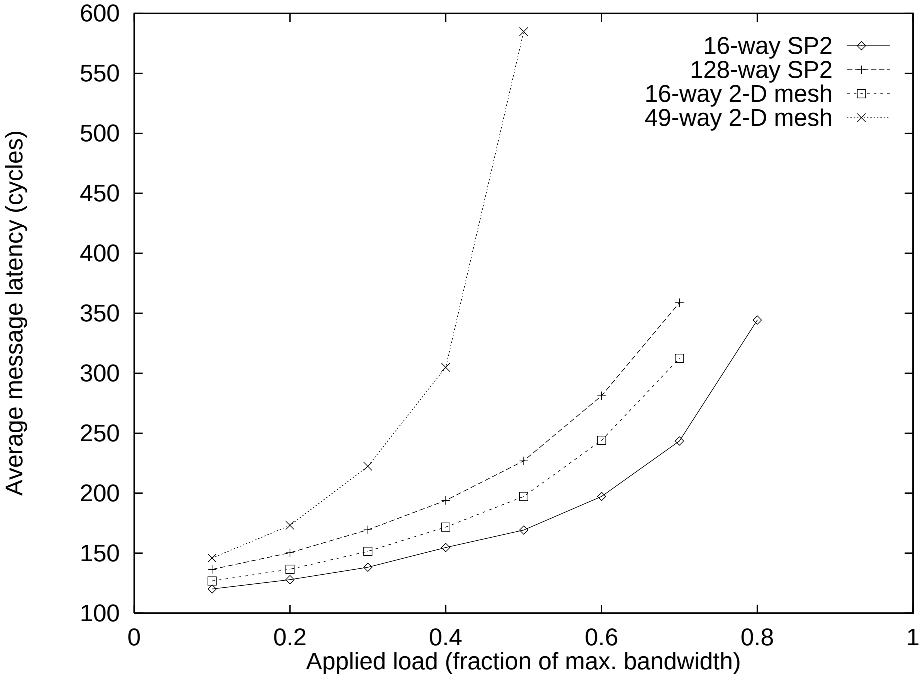

Figure 12: Average message latency for 100-byte messages

dimension-order routing[5] for the meshes [2]. Figure 12 displays the latency results for 100-byte messages. The 16-way systems achieve lower latencies until saturation, and saturate at higher input load than the large systems. The 49-way mesh system saturates much lower load than the 128-way SP2, because it has limited bisection bandwidth. A 128-way mesh would perform significantly worse than the 49-way mesh for random loads. The higher large system latencies are partly a result of the increased number of average hops to reach a randomly chosen destination. In the 16-way SP2 topology, packets traverse 3 or less switching elements, while the 128-way contains 6-level paths. The increased levels have two effects: (a) an addition of about 5 cycles of minimum latency per hop, which is the major difference at the 0.1 input load, and (b) more sites at which to encounter packet contention, a factor which increases with system load.

Figure 13 shows 500-byte message simulations which provide qualitatively similar results. These larger messages are more stressful on the SP2 network (or on any network) because the randomness of the traffic within localized areas of the network is effectively reduced, presumably causing hot spots to spawn more often and to last longer. Again, the smaller

[5]This is a common way of avoiding mesh or torus deadlock. Packets travel first in the x direction, and then in the y direction.

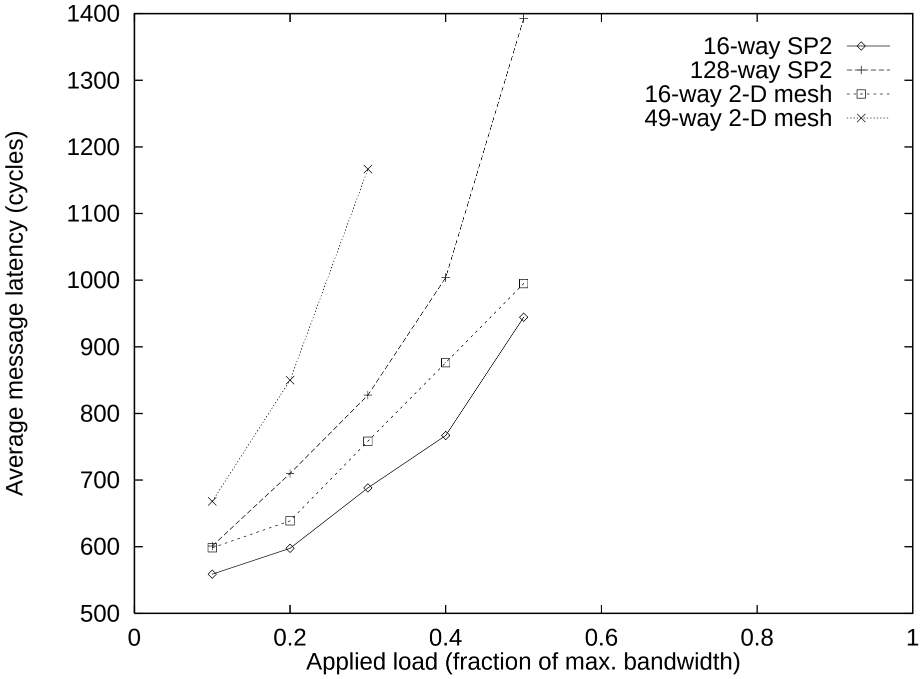

Figure 13: Average message latency for 500-byte messages

networks achieve lower latency for lighter loads, but saturate in the same bandwidth range, bolstering the claim of scalable bandwidth.

Network performance above the saturation load is also of interest. Average latency is not meaningful above saturation, however, because packet queuing time at the source goes to infinity when the network cannot dispose of packets as fast as they arrive. Output bandwidth, though, remains valid, and indicates maximum sustainable bandwidth. Figure 14 plots received traffic versus input traffic for 500-byte messages on SP2 networks and meshes. Until saturation, each line is straight with a slope of 1. After saturation, output bandwidth levels off. The figure shows a dramatic difference between the saturation bandwidths of the 49-way mesh and the other systems, yet the 128-way SP2 topology maintains almost as much bandwidth as the 16-way SP2.

To summarize, the characteristics of the Vulcan switch chip and the SP2 topology choices combine to provide systems with near linear bandwidth scaling for a large range of system sizes.

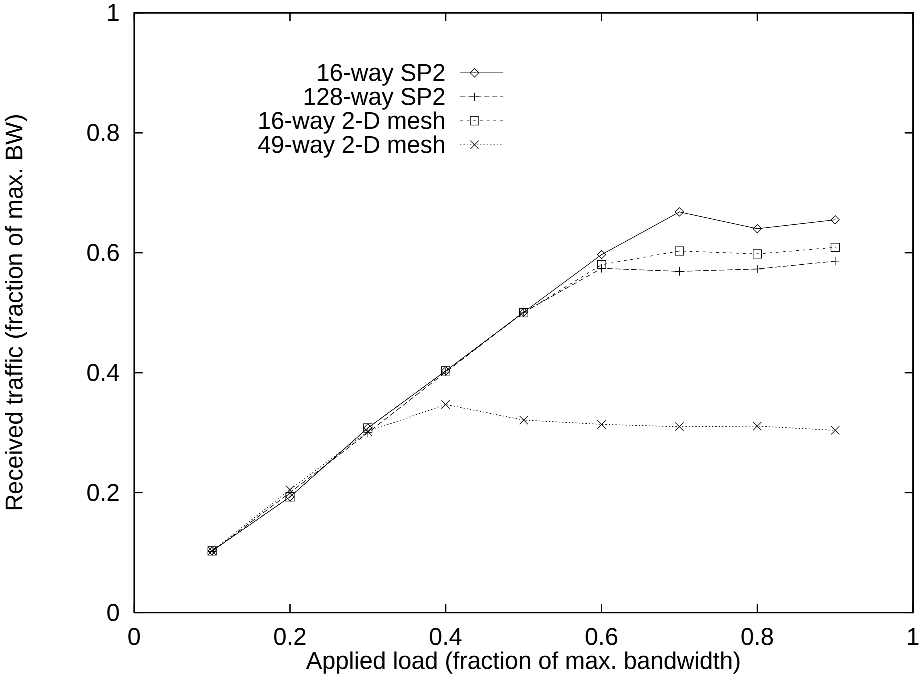

Figure 14: Output bandwidth for 500-byte messages

7 Support Software

Switch service operations provide functions such as initialization of switch chips and adapters and determination of acceptable packet routes. For the SP2 Switch network, these service operations use the same links used for packet data transfer. The data and service activities occur during different *modes* of Switch operation, known as run mode for data transfer and service mode for service. The entire network can be scheduled to synchronously move into and out of service mode [6]. The mechanism for achieving these mode changes will not be detailed here.

The software that explores and initiates the Switch and controls the Switch during network service is called the "Worm" for reasons that will be apparent shortly. Another crucial element of the service software is the Route Table Generator (RTG). RTG, using information on the network state gathered by the Worm, calculates packet routes for each active source-destination pair of nodes in the system.

This section introduces the protocol used during service mode, and then overviews the Worm and RTG software.

Service mode protocol

The protocol described in §2, used for passing data from source nodes to destination nodes, is optimized to minimize packet forwarding delay. However, this protocol contains no mechanism for addressing the individual devices in the network—nodes are always the intended destination.

A *circuit-switched* protocol is used during service mode to communicate with network devices. In circuit switching a path is first set up by a control packet or packets, and succeeding packets traverse that configured circuit until further control packets change the circuit. We will use the terms service packet and service message interchangeably, because service messages are constrained to be no longer than one packet ($\leq$ 255 bytes) in length. Another characteristic of service mode is that there is only one node allowed to send the control packets that configure the network, to avoid the complexity of managing conflicting circuits in hardware. This *service node* is arbitrarily selected.

Circuit-switching is advantageous for servicing network devices. First, typically, once a circuit is configured to a particular device, many service messages are passed to and from that device. In fact, a circuit is often configured as a loop beginning at the service node, going to a network device such as a switching element, and returning to the service node. This circuit allows the service node to quickly verify that the service message was successfully executed, and can also be used to return information—substituted into the packet—from the destination device.

Secondly, multicasting or broadcasting circuits are easily constructed with little additional complexity in the switching hardware with the "one service node" assumption. The Vulcan switch chip implements these multicasting circuits by allowing an incoming service packet to be forwarded to an arbitrary set of output ports (configured by previous control packets).

The Worm

A major component of the SP2 Switch network service software is called the Worm. It executes on the service node (also called the primary node). The Worm software handles Vulcan switch chip and adapter initialization, channel tuning [6], global time synchronization [7], fault determination, and diagnostic services for the Switch network. The Worm executes only in service mode, controlling and observing the network by sending and receiving service messages.

The Worm is designed to handle arbitrary bidirectional topologies. This provides flexibility for special customer requests and allows special-purpose configurations for diagnostic purposes. An important consequence: the non-uniformities caused by faulty network components are accommodated easily.

The Worm software is guided by a file that describes the expected topology. It searches the network in a breadth-first manner, entering status information about each device and link in an internal database. This database is subsequently used by the Route Table Generator to generate valid node-to-node routes.

Route generation

SP2 packet routing is source-based: the source node places route flits into each packet, and the switching elements of the network obey these directives. For each node, a route table contains valid routes for each destination. This route table is updated during system initialization and after network status changes.

The SP2 routing algorithm, called the Route Table Generator (RTG), calculates the routes required for node-to-node communication over the SP2 Switch network. The RTG is based on an SP1 routing algorithm that provides a single shortest path between each pair of processor nodes [19]. The shortest path approach corresponds to traveling from a source to a "least common ancestor" and back to the destination, as described in §5. We enhanced the SP1 algorithm for SP2 to generate multiple (4) paths between each node pair.

The SP2 Switch network provides a rich set of paths between node pairs. In such networks, the selection of the routes is important because of its impact on performance. In an attempt to prevent congestion in the network, the RTG selects routes that traverse the links and switches in node-to-node paths in a balanced manner. Furthermore, system software uses the RTG generated multiple routes between each node pair in a round robin fashion to more uniformly utilize the network.

The RTG selects routes only from (deadlock free) shortest paths. Shortest paths minimize the resources used by packets, reducing network congestion. The RTG uses a modified breadth-first search algorithm for building a spanning tree rooted at each source node, and then follows the spanning tree paths to find the shortest paths from the source node to the rest of the processor nodes. We added a simple static load balancing strategy to ensure that links and switches are included in the selected routes in a balanced manner. The RTG maintains a usage counter for each switch chip output port. The counter indicates how many times the port has been used during route generation. While building a spanning tree from a given source node, each time a source–destination path is found, the counter is incremented for each output port in the path. The usage count of the ports determines the breadth first search order such that, from a given switch, the RTG algorithm first visits the switches connected to the least frequently used output ports [19].

The RTG routes are stored in a route table in each processor's memory. The route table approach enables routing to be done in a topology independent fashion which is important for scalability and fault-tolerance. Larger networks of various topological properties can be implemented easily without having to change the switch hardware or the routing algorithm, and the RTG routes around the missing links and switches reported unacceptable by the Worm.

8 Conclusion

The SP2 High-Performance Switch is a low latency, high bandwidth switching network that can scale aggregate bandwidth for systems containing hundreds of processing nodes. The network is a bidirectional MIN, and provides at least 4 usable redundant paths for most pairs of communicating nodes. Data and service operations share a single network. The network is synchronous, with mechanisms for achieving a closely synchronized global time visible to each node. The basic switching element is the Vulcan switch chip, which contains a unique central buffering scheme for reducing the impact of network contention. The programmable SP2 processor communication adapter provides high bandwidth to and from a processor and provides the processor with opportunities to overlap communication and computation.

Acknowledgements

The inspiration for, and the design and implementation of, the SP2 communication subsystem hardware is due to numerous others from IBM Research and IBM's Highly Parallel Supercomputing Systems Lab. In particular, we cite the contribution of Gerard Salem and Singpui Zee to the SP2 adapter design. Harish Sethu made crucial contributions to the choices and understanding of SP2 topologies for larger systems. Monty Denneau was one of the original architects of the Vulcan machine upon which the SP2 communication subsystem is based.

References

[1] D. A. Reed and R. M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing.* Cambridge, MA: The MIT Press, 1987.

[2] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, vol. 39, pp. 775–785, June 1990.

[3] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, pp. 194–205, March 1992.

[4] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communications switching technique," *Computer Networks*, vol. 3, pp. 267–286, Sept. 1979.

[5] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, pp. 558–565, July 1978.

[6] 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*, pp. 150–157, May 1994.

[7] B. Abali and C. B. Stunkel, "Time synchronization on sp1 and sp2 parallel systems," in *Proc. 9th Int. Parallel Processing Symp.*, April 1995.

[8] Y. Tamir and G. L. Frazier, "Hardware support for high-priority traffic in VLSI communication switches," *J. of Parallel and Distributed Computing*, vol. 14, pp. 402–416, April 1992.

[9] 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.*, pp. 268–274, April 1994. (An extended version of this paper is also available as Research Report RC19492 from the IBM T. J. Watson Research Center, Sept. 1993.).

[10] M. Snir, P. H. Hochschild, D. Frye, and K. Gildea, "The communication software and parallel environment of SP2," *IBM System Journal*, this issue.

[11] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. on Computers*, vol. C-24, pp. 1145–1155, Dec. 1975.

[12] M. C. Pease, III, "The indirect binary n-cube microprocessor array," *IEEE Trans. on Computers*, vol. C-26, pp. 458–473, May 1977.

[13] I. D. Scherson and C.-H. Chien, "Least common ancestor networks," in *Proc. 7th Int. Parallel Processing Symp.*, pp. 507–513, 1993.

[14] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Trans. on Computers*, vol. C-34, pp. 892–901, Oct. 1985.

[15] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak, "The network architecture of the Connection Machine CM-5," in *Proc. 1992 Symp. Parallel Algorithms and Architectures*, pp. 272–285, ACM, 1992.

[16] W. Gropp and E. Lusk, "Early experiences with the IBM SP1 at Argonne National Laboratory," *IBM System Journal*, this issue.

[17] R. Agarwal, Alpern, Carter, F. Gustavson, Klepacki, R. Lawrence, and Zubair, "High performance parallel implementation of NAS kernel benchmark on SP2," *IBM System Journal*, this issue.

[18] V. Naik, "An implementation of NAS parallel benchmark on distributed memory systems," *IBM System Journal*, this issue.

[19] B. Abali and C. Aykanat, "Routing algorithms for IBM SP1," in *Proc. Parallel Computer Routing and Communication Workshop*, (Seattle, WA), May 1994. To appear in Springer-Verlag Lecture Notes in Computer Science. Also available as IBM Research Report RC19616.