Performance of the SCI Ring

Steven L. Scott, James R. Goodman and Mary K. Vernon

Department of Computer Sciences
University of Wisconsin - Madison

Abstract

The Scalable Coherent Interface (SCI) is an emerging IEEE standard that provides computer-bus-like services to a set of nodes via fast, unidirectional links. This paper presents the first detailed performance study of the SCI ring, using both analytical models and simulation. Performance is analyzed for uniform and non-uniform traffic, and the effect of the ring's flow control protocol is studied.

The queueing model is based on an M/G/1 queue, augmented to include the effect of packet trains on the mean and variance of the source transmission time. The model is validated against simulation results, and shown to be quantitatively accurate for uniform workloads, and at least qualitatively accurate for non-uniform workloads. The flow control mechanism is shown to effectively prevent node starvation and reduce the ability of nodes to unfairly consume ring bandwidth, but at the cost of decreased overall ring utilization. The SCI ring is also compared to a standard bus, modeled with a simple M/G/1 queue, and shown to provide substantially higher throughputs and lower latency than a bus with realistic clock speeds.

1. INTRODUCTION

The Scalable Coherent Interface (SCI) is a proposed IEEE standard that provides very-high-performance, bus-like functionality to a large number of processor nodes [IEEE91, Gust92]. Using a packet-based communication protocol based on unidirectional links connected in a ring, it provides a shared-memory interface, including cache coherence, to the nodes. The protocol has been developed over a period of approximately four years, has included participation by representatives of dozens of companies and many universities and has assembled appropriate expertise in many different disciplines to solve the plethora of problems associated with a novel design.

The SCI includes protocols at three different levels: the physical level, the logical level, and the cache-coherence level. The logical level provides the protocol for reliably transmitting packets between nodes. The node interface consists of two unidirectional links, an input and an output, which are used to connect nodes together in the basic topology of a ring. The ring can in theory be arbitrarily large, but performance considerations lead to the expectation that a ring will be limited to a modest number of processors, numbering at most a few dozen and perhaps as few as two. Larger systems can be built by connecting together multiple rings by means of switches, that is, nodes containing more than a single interface.

The ring is unusual in that each node provides a bypass buffer capable of storing temporarily a packet arriving from its upstream neighbor while it is transmitting a packet. This buffer allows nodes to transmit concurrently rather than having to wait for a token, but results in long latency if all nodes happen to initiate transmission simultaneously on an idle ring. Because of the novel construction of the ring and the attendant clock rates achievable in the design, very high performance is expected, and a peak bandwidth of one gigabyte per second is easy to demonstrate. The nature of the protocol, however, makes both the achievable bandwidth and the observed latency much harder to predict. Reported here is the most detailed study to date attempting to analyze the performance of a single ring. It includes analyzing the performance under a variety of conditions, including the number of nodes in the ring, the service rate of queues, the size of packets, and the distribution of sources and destinations for the packets. In addition, a mechanism to assure fairness in the ring is investigated to assess its impact on the performance. The SCI protocol does provide priority scheduling, but this aspect of the protocol is not investigated in this study. The ring is studied both by the use of an analytical model and through simulation.

The remainder of the paper is organized as follows. Section 2 of the paper describes the protocol of the SCI logical layer. Section 3 describes the analytical model. Section 4 presents and analyzes the results of the study, and Section 5 summarizes the conclusions.

2. THE SCI LOGICAL-LEVEL PROTOCOL

The key idea behind the SCI logical level protocol is the use of unidirectional, point-to-point links that can be clocked at a rate independent of the signal latency between nodes. The basic building block of an SCI system is a *ring* (sometimes called a *ringlet*) of two or more nodes connected by these links. The protocol is designed such that short of an actual hardware failure packets are guaranteed to be accepted by each node that they pass through as they traverse the ring. Therefore, a node can output a *symbol* of information on every clock cycle, and there is *no* direct feedback from a node to its upstream neighbor. A packet might not be accepted by its *destination*, however, due to queue congestion. The protocol uses packet-level acknowledgements to deal with this issue.

2.1. Basic Protocol

This section presents a summary of the basic protocol. Details such as ring initialization and error detection/recovery are not covered, nor is all the functionality of the standard presented. Buffer management is somewhat simplified; we assume a single transmit and receive queue per node, whereas the actual system requires dual queues in order to support a higher level protocol. The cache coherence level of the SCI standard is not considered at all. Much more detail can be found in the standard [IEEE91].

A packet traversing an SCI ring is sent from a *source* node to a *target* node in the form of a *send packet*. The target node then *strips* the send packet, and returns an *echo packet* around the remainder of the ring. This echo packet tells the source node whether or not the send packet was accepted by the target. If the packet was not accepted, then the source must retransmit it.

A send packet consists of a 16 byte header and an optional data component of up to 256 bytes. The header contains

© 1992 ACM 0-89791-509-7/92/0005/0403 $1.50

command and control information, a 16-bit CRC (Cyclic Redundancy Check) and a 64-bit memory address (16-bit node id and 48-bit intra-node address). We assume a data component size of 64 bytes, which corresponds to the SCI cache line size. Echo packets are 8 bytes long. We assume a 16 bit link width (the standard defines both a 16 bit copper implementation and a serial, fiber optic implementation)

Figure 1 shows a block diagram of an SCI node and ring interface. A node transmits a symbol onto its output link on every SCI cycle. When a node has no packet to transmit, it sends an *idle symbol*. When a source desires to send a packet over the ring, it places the packet in its transmit queue. If the ring buffer is empty and the node is not currently transmitting a packet from the stripper, the send packet is immediately output onto the ring. When a source packet is transmitted, a copy must either must be saved at the head of the queue (thus blocking further transmissions) or placed into an optional *active buffer*. The copy is either discarded or used for retransmission when the echo packet is received.

Upon arrival at the downstream node, a send packet is parsed and either stripped or *passed* along the ring. In the absence of contention, a passing packet may be routed directly from the *stripper* to the output link. If the ring buffer is not empty or the transmit queue at the node is currently transmitting a packet, the passing packet is routed into the ring buffer. If a passing packet and a source packet are ready to transmit on the same cycle, the transmit queue is given priority and the passing packet is routed to the ring buffer.

When the transmit queue is done transmitting a packet, if the ring buffer has accumulated any symbols, output resumes from the ring buffer (which may still be receiving symbols from the stripper). This is known as the *recovery stage*, and lasts until the ring buffer is completely emptied. The node is not allowed to transmit another source packet during the recovery stage. To empty the ring buffer, the node either must see gaps in the stream of incoming packets, or create gaps in the packet stream by stripping packets for which it is the target. During these gaps, the buffer can be drained while not being simultaneously filled. If the ring buffer is empty after a source transmission completes, then there is no recovery stage.

When a send packet reaches its target, it is stripped and either placed into the receive queue (space permitting) or discarded. The node uses the bandwidth created by stripping the packet to insert idle symbols, transmit symbols from the transmit queue or drain the ring buffer. The last four symbols of the send packet are replaced with an echo packet that continues its way around the ring to the packet's source. At the source, the echo packet is matched with a saved send packet in an active buffer or at the head of the transmit queue, and the appropriate action is taken (discarding or retransmitting the send packet).

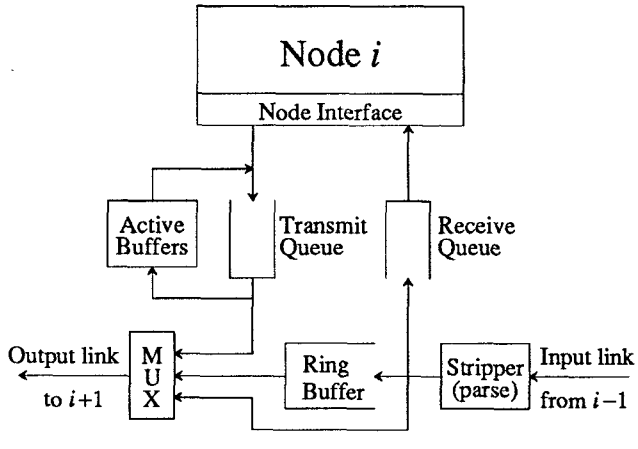


Figure 1: An SCI Node

One last feature of the protocol that needs mentioning is that packets are always separated by at least one idle symbol. This allows the stripper to periodically delete an idle symbol, if necessary to adjust for a slowly varying clock period between neighbors (this is known as *elasticity*). It also assures timely distribution of priority and other information carried in the idle symbols. We do not consider elasticity or priorities here, but *do* require the intervening idle symbols. For the purposes of the basic model, this is equivalent to increasing the length of all packets by one symbol.

2.2. Flow Control

The basic protocol described above works fine for uniform traffic rates and routing distributions. However, it allows for nodes to be unfairly starved in the presence of certain non-uniform traffic patterns. Consider a node that partially fills its ring buffer during a transmit queue transmission. If the node then receives a continuous stream of passing packets, then its recovery stage can take arbitrarily long, denying it the chance to transmit another packet. For this reason, the SCI protocol includes a flow control mechanism that uses *go bits* in the idle symbols to enforce an approximate round robin ordering under heavy loads. The flow control mechanism is complicated by a priority mechanism that partitions the ring's bandwidth between high and low priority nodes. While the priority mechanism has certain special uses, such as in real-time systems, it is not likely to be used for general purpose multiprocessors. We assume that all nodes have equal priority, and present the simpler flow control mechanism that results.

Each idle symbol contains a go bit which is either set (making it a *go-idle*) or cleared (making it a *stop-idle*). The stripper passes all idles and passing packets (as well as echos for packets that it strips) to the transmitter stage of the node interface. When it strips a packet, it fills the empty slots with idle symbols. When a node is not transmitting a packet from its transmit queue and is not in the recovery stage, it simply passes all symbols -- send, echo and idle -- from the stripper to its output link. Whenever the transmitter emits a go-idle, it continues to emit go-idles until the next packet boundary, possibly converting passing stop-idles into go-idles (this is called go-bit *extension*).

A node may *only* transmit a source packet immediately following a go-idle. During transmission of a packet, a node maintains the inclusive-OR of all go bits it receives from the stripper. If the ring buffer does not fill up at all during transmission, then the node postpends an idle symbol to its packet using the saved go bit it maintained during the transmission, and then continues to either transmit another source packet or output symbols from the stripper.

If the ring buffer *does* fill up at all during transmit queue transmission, then the node enters the recovery stage. All idles sent during the recovery stage, including the idle postpended to the original source transmission, are stop-idles. The node continues, however, to maintain the inclusive-OR of go bits it receives throughout the recovery stage. When the recovery stage ends (the last symbol is drained from the ring buffer), the saved go bit is released in the postpending idle just as it was for the postpending idle of a source transmission when the recovery state was not entered.

The stop idles that are transmitted during a recovery stage inhibit the downstream neighbors from sending new packets and eventually provide enough slack for the node to drain its ring buffer and send a packet. In the absence of contention, all idles on the ring will be go-idles, and a newly arriving send packet can always be sent immediately.

3. ANALYTICAL MODEL

This section presents an analytical performance model of the SCI ring. The model is useful for a variety of reasons: it allows the quick exploration of a large state space, it's precisely defined and can be implemented by other researchers and engineers, and it helps us gain insight into the processes being modeled. Results from the model, as well as from detailed simulations, will be presented in section 4.

The model is based upon an approximate, iterative solution of the M/G/1 queue [Klei75] (see Figure 2). It does not consider flow control, limited active buffers or target queue overflow. The effect of these factors can be determined from simulation results. The model does consider, and effectively deals with, ring buffer fill-up during transmit queue transmissions, the transmission recovery process, the formation and effect of packet trains[1], non-homogeneous packet arrival rates and non-uniform packet destination probabilities, delays due to queueing in ring buffers and variance of transmit queue service times. We present only a summary of the model here, highlighting the important aspects of the approach.

3.1. Model Inputs

Inputs to the model are the ring size (N), packet arrival rates (λ_i), routing probabilities (z_{ij}), packet lengths (l_{addr}, l_{data}, l_{echo}), packet type ratio (f_{data}, f_{addr}), transmission delay (T_{wire}) and parsing delay (T_{parse}). Note that each node has a distinct arrival rate and a distinct, possibly non-uniform packet destination probability distribution.

3.2. Discussion of Model Equations

The model equations are given in Appendix A. Equations (1) through (14) are straightforward. In the first twelve equations quantities such as mean and variance of packets lengths, link utilizations, and various throughputs and ratios are derived directly from the inputs. Note that packet lengths include the idle symbols. The model then ignores these idles and considers only the remaining "free" idles.

Two key assumptions in the model are that the number of idle symbols between packet trains is geometrically distributed and that packet trains contain a geometrically distributed number of packets[2]. Thus the probability than an idle symbol is directly followed by a packet, $P_{pkt,i}$, is the inverse of the mean time between packet trains (Equation (15)), and the expected remaining length of a passing packet train that is interrupted by the arrival of a packet from the transmit queue is equal to the mean length of a packet train (first occurrence of $l_{train,i}$ in Equation (16)).

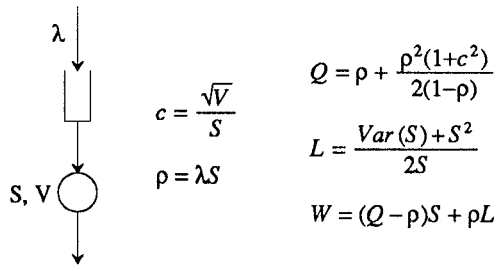


Figure 2: The M/G/1 queue

λ - arrival rate; S - mean service time; V - variance of service time; c - coefficient of variation; ρ - utilization; Q - mean queue length; L - mean residual life; W - mean wait time

The transmit queue service time includes the recovery period, which lasts until the ring buffer is empty. Thus total service time is equal to the time that would elapse if we were to simply discard the send packet and wait until a number of idle symbols equal to the length of the packet passed through the node. The first half of Equation (16) accounts for the possible residual life of a passing packet train when a send packet arrives to an idle transmit queue. The second half accounts for the remaining time taken to observe the required number of idle symbols. After each of the idle symbols, another packet train passes through with probability $P_{pkt,i}$.

Equations (18) - (22) compute new estimates of coupling probabilities. The second term of Equation (18) considers new couplings that are formed when a source packet is injected at node i. Equation (19) calculates the mean number of coupled packets that enter the stripper at node i for each packet stripped, where stripped packets include echo packets that are consumed and send packets that are converted into echo packets. Equation (21) considers couplings from the upstream neighbor that are destroyed by the stripper at node i. Equation (22) computes the net effect of stripping packets and removing couplings. Note that we assume in these equations that coupling probabilities are not correlated with packet length. The relation between service time and coupling probabilities is cyclic. The equations are solved iteratively until the coupling probabilities converge.

After the above convergence, several metrics can be computed. The variance of the packet train length (Equation (24)) is computed using the geometric distribution of the number of packets in a packet train. The variance of the transmit queue service time (Equation (26)) is computed using the binomial distribution of the number of packet trains arriving during the transmission/recovery period. The variance calculation in Equation (26) also assumes that the total recovery time is a constant (Equation (25)) times the last term of Equation (16). That is, we assume a strong correlation between the delay due to packet trains arriving during the recovery period (the last term of Equation (16)) and the residual life of a passing packet train (the first half of Equation (16)). Finally, the mean backlog seen by a passing packet at node i (Equation (32)) is computed by dividing the total backlog created by an injected packet, by the mean number of passing packets per injected packet.

Experience implementing this model has shown that convergence is faster for smaller ring sizes. We required average change in coupling probabilities to be less than 10^{-5} for convergence. Approximately 10 iterations were needed for N=4, 30 for N=16 and 110 for N=64. Total time to solve the model for N=64 on a DECstation 3100 is about 1 second. Comparable simulation time (simulating 9.3 million cycles, as was done for this paper) is over 4 hours.

4. RESULTS

This section presents results derived from both the analytical model and a detailed, parameter-driven simulator of the SCI ring. The inputs to the model and to the simulator are identical. The ring is modeled as an open system (Poisson arrivals), with the arrival rates, packet lengths, mix of packet types, routing probabilities, ring size, wire transmission delay and packet parsing delay specified as inputs. The simulator has the additional ability to consider flow control and limited buffer space (active buffers and receive queues). Since the ring is modeled as an open system, latency becomes infinite as saturation is reached. An actual system, of course, would have a limit to the number of queued or outstanding requests, and nodes would be stalled at some point rather than continuing to add requests. Section 4.6 illustrates the component of total delay due to queueing.

The unit of length in the model and simulator is one link width, and the unit of time is one clock cycle. We assume (as per

[1] Note that these packet trains arise from collisions between passing packets and source packets as well as from the insertion of trains at source nodes.

[2] We comment on the accuracy of these and other model assumptions in section 4.9.

the standard) a 16-bit link with a 2 ns cycle time. Using these assumptions, we present output latencies in *ns* and throughputs in *bytes per ns*. Throughputs are calculated using the entire packet, including address, command and control information. Section 4.7 considers sustained data throughput using a read request/read response model.

Many other parameters have been fixed or limited in order to make the problem space tractable. Since the number of nodes in a ring is expected to be small, we analyzed ring sizes of 4 and 16 nodes. Except where noted, we assumed that 60% of send packets were address/command only (16 bytes), and 40% included data blocks (80 bytes) (we refer to these as address packets and data packets for the remainder of the paper). This corresponds to a workload in which most of the traffic consists of paired address and data packets. We assume a fixed minimum delay of 4 cycles per node traversed by a packet: one cycle to gate a symbol onto an output link, one cycle for the symbol to reach its downstream neighbor and two cycles to parse a symbol before routing it to the local node or to the next output link. Message latencies also include one cycle to originally queue the packet, and a delay equal to the packet length to consume the packet as it arrives at the target node. We assume unlimited active buffers at each node, but only one or two active buffers are actually needed to approximate this [Scot91].

The simulator implements the protocol described in section 2 on a cycle by cycle basis, explicitly tracking each symbol on the ring. Simulations were run for 9.3 million cycles each, and 90% confidence intervals were computed using the method of batched means. Confidence intervals were generally under or about 1%, except near saturation, where they sometimes increased to a few percent.

4.1. Uniform traffic

Figure 3 shows the performance of 4- and 16-node SCI rings with uniform arrival rates and routing probabilities and no flow control. Each graph includes three sets of data, one with all address packets, one with all data packets and one with 40% data packets. Both simulation and model results are shown. The model is very accurate for the 4-node ring. For the 16-node ring, the model is accurate for the all-address-packet workload, but underestimates latency under moderate to heavy loading for the other workloads. Even for the worst case, however, the model provides a good estimate for the behavior of the ring. The reason for the error is identified and discussed in section 4.9.

Throughput is higher for the workload with larger packet sizes. There are two reasons for this. First, a smaller proportion of the ring bandwidth is used for the idle symbols that must separate each packet (we include only bytes within packets in the throughput measure). Second, the bandwidth consumed by echo packets becomes smaller relative to the bandwidth used by send packets. Throughput could also be increased by use of packet locality. Unlike a shared bus, a ring requires less bandwidth if the packets are sent a shorter distance (message latency is similarly reduced). For the purposes of this paper, we assume equally distributed destinations.

Figure 4 illustrates the effect of flow control on uniform traffic for ring sizes of 4 and 16. Each graph includes two sets of data, one with all address packets, and one with all data packets. Results for the mixed address/data workload fall in between these. We can see that even with uniform traffic loading, flow control significantly reduces the maximum throughput. The reason for this is that there are times when a node cannot transmit a source packet, even though there are available slots in which to do so, because another node has stopped sending go bits in order to clear its ring buffer. The degradation is greater for the 16-node ring than for the 4-node ring. Simulations indicate that the throughput degradation from flow control is greatest for ring sizes in the 10 to 20 range, and actually lessens slightly for larger rings [Scot91].

4.2. Node Starvation

This section examines the situation in which a node is inhibited from transmitting by reducing the number of breaks it sees in its pass-though traffic. Figure 5 presents the performance for 4- and 16-node rings where all nodes are routing uniformly, except that no packets are routed to node 0 (the starved node). Mean message latencies are plotted for individual source nodes (labeled P0, P1, *etc.*). In Figure 5(a), we see that P0 saturates before the other nodes. As the throughput per node reaches about 3.2 bytes/node/ns, P0's arrivals can no longer be satisfied and its message latency goes to infinity (recall that this is simulated as an open system). As P1, P2 and P3 increase their throughput beyond this point, the realized throughput of P0 is actually driven back down to 0. This causes the unusual shape in the curves for P1 and P2. P1, P2 and P3 all reach the same saturation bandwidth.

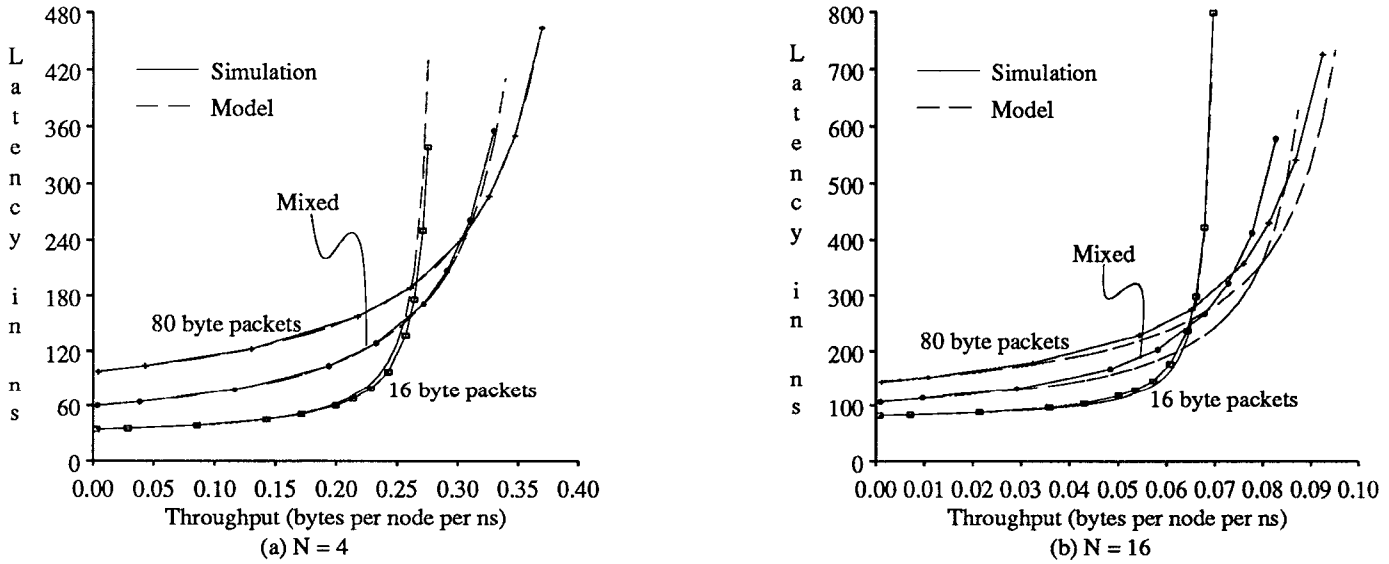


Figure 3: Uniform traffic without flow control

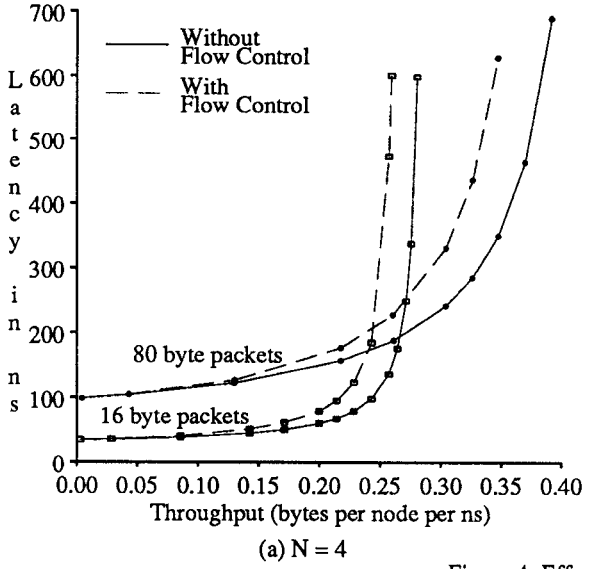
The equations in the model assume that the system is not in saturation, so in order to model the behavior after P0 has saturated, the model detects saturated queues, and automatically throttles back the corresponding arrival rates to keep the transmit queue utilization at exactly one. The model qualitatively predicts correct behavior, including the throttling of P0's throughput and the corresponding inflection points in the P1 curve. However, the model underestimates the impact of the non-uniform traffic, and quantitative error is fairly large when the starved node is in saturation.

For a ring size of 16 (Figure 5(b)), the disparity between nodes is not as pronounced. The starved node reaches almost as high a bandwidth as the other nodes before it saturates. This is because the non-uniform routing causes smaller differences in link utilizations for the larger ring. The model correctly predicts the spread in performance between the starved node (P0) and the least affected node (P15). The absolute error, however, is fairly significant under heavy loads.

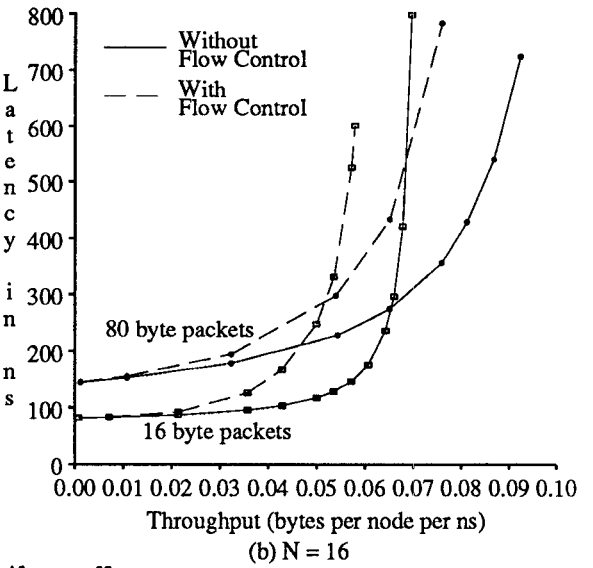
Figure 6 demonstrates the effect of flow control on node starvation. Parts (a) and (b) show the message latency for each node as the traffic is varied. In parts (c) and (d), the ring is in saturation (all nodes are trying to send as often as possible), and the realized throughput for each node is shown.

In Figure 6(a), we see that the addition of flow control reduces the disparity between the performance of the four nodes, but at an overall reduction in throughput. The throughput of P0 is not driven back down by the other nodes, as it is without flow control. Note, however, that the performance is not fully equalized; P0 achieves a smaller maximum throughput than P1, P1 achieves a smaller maximum throughput than P2, *etc.*

Figure 6(c) shows the saturation bandwidths for the 4-node ring with P0 still being starved. Without flow control, P1, P2 and P3 all achieve the same throughput, but P0 is *completely* starved.

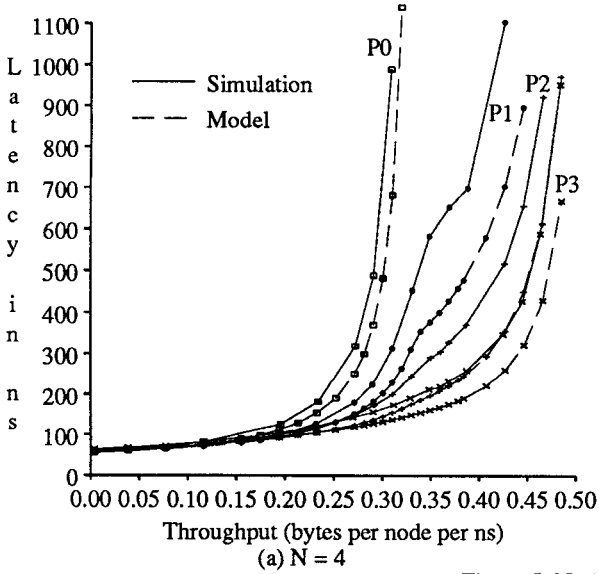


(a) N = 4

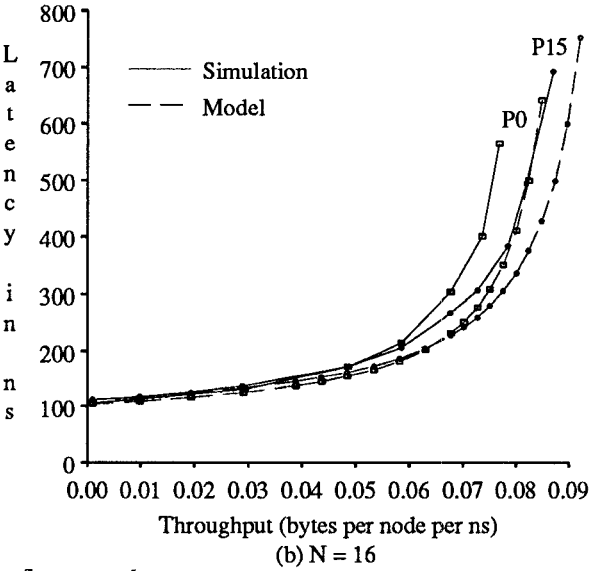


(b) N = 16

Figure 4: Effect of flow control on uniform traffic



(a) N = 4



(b) N = 16

Figure 5: Node starvation without flow control

Because the ring is fully utilized and it is not receiving any packets, it has no opportunities in which to transmit a packet (*i.e*: it enters an infinite recovery stage). The flow control mechanism successfully deals with this problem. With flow control, the total ring throughput is reduced slightly, and the throughput of the non-starved nodes is reduced significantly, but the starved node is no longer kept from transmitting. The flow control mechanism does not achieve full equal partitioning of bandwidth, however. The throughput of a node is limited by how quickly it can empty its ring buffer after a transmission, and the flow control protocol guarantees that even a starved node will make forward progress in the recovery stage by throttling downstream transmissions and thus creating gaps in its incoming packet stream. However, a node whose pass-through traffic is lower overall (due to non-uniform traffic) will still be able to recovery more quickly on average.

Figure 6(b) shows the effect of flow control for a ring size of 16 with P0 being starved. The addition of flow control almost completely equalizes the performance of the various nodes (again at an overall reduction in ring capacity). Figure 6(d) shows the saturation bandwidths for the 16-node ring. Although the impact on P0 was small under light to medium traffic, P0 is completely starved when the ring is fully loaded. Flow control reduces the realized throughput of the non-starved nodes and allows the starved node to transmit. The bandwidth is much more equally divided than it was for the 4-node ring. This is because the differences in link utilizations caused by the non-uniform traffic are smaller for the larger ring.

4.3. Hot Sender

In this section, we examine the ring's behavior in the presence of a "hot sender" (a node that attempts to use as much ring bandwidth as possible). Figure 7 presents the performance for 4- and 16-node rings where packet destinations are uniformly distributed, but node 0 always wants to transmit a packet. P1, the first downstream node from the hot sender, is severely affected by the extra traffic. The hot node degrades the performance of all other nodes on the ring, affecting the closest nodes more heavily.

The model is very accurate for a ring size of four (Figure 7(a)). For a ring size of 16 (Figure 7(b)), the model is qualitatively accurate, but slightly underestimates latency for most of the nodes, and significantly overestimates the latency for the immediate downstream neighbor of the hot node (the reason that the model overestimates P1's latency is actually that it underestimates P0's latency (the hot node), allowing P0 to send more than it would in reality).

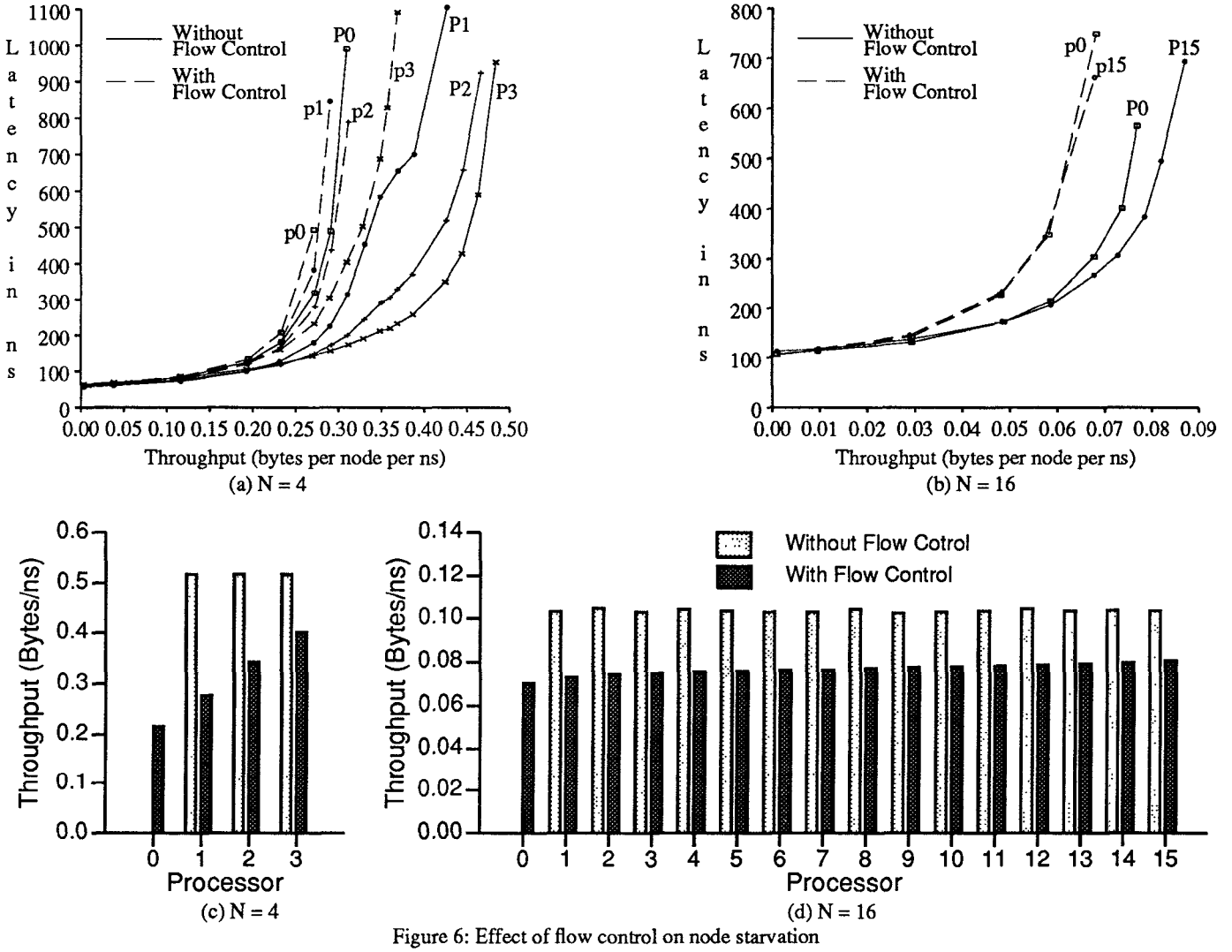


Figure 6: Effect of flow control on node starvation

Figure 8 demonstrates the effect of flow control on a hot sender. Parts (a) and (b) show the message latencies for each node as a function of throughput. The addition of flow control equalizes the effect of the hot node on the message latency for the other nodes on the ring. Performance is improved for some nodes and degraded for others, but the hot node's downstream neighbor is no longer severely penalized.

This phenomenon can be clearly seen in parts (c) and (d), which show vertical slices of the throughput-latency curves under moderate throughput from the "cold" nodes (0.194 bytes/ns in Figure 8(c), 0.048 bytes/ns in Figure 8(d)). Without flow control, the mean message latencies experienced by the cold nodes vary significantly, with the closest downstream nodes being affected the most. With flow control, the hot node affects all other nodes approximately equally. The nearest downstream node, in particular, is no longer subjected to extremely large latencies. The improved ring fairness is achieved at the expense of the hot sender's throughput. Without flow control, it realizes a rate of 0.670 bytes/ns on the 4-node ring. With flow control, it realizes only 0.550 bytes/ns. For the 16-node ring, the hot sender's throughput is reduced from 0.526 bytes/ns to 0.293 bytes/ns. For certain applications, most notably real-time systems, it may be desirable to allow one node or a set of nodes to consume more than their share of ring bandwidth. SCI provides a priority mechanism to satisfy this requirement.

In addition to hot senders and node starvation, we have examined producer-consumer and other non-uniform workloads. Though not presented here, the results are similar. The flow control mechanism reduces the effects of greedy nodes on the rest of the ring, and provides all nodes with a reasonable approximation to their share of the bandwidth, regardless of the non-uniformities present in the communication pattern.

4.4. Comparison to a Conventional Bus

This section compares the SCI ring to a conventional, synchronous bus. The prime advantage of SCI, at the logical-layer level, is its use of fast, point-to-point links. The unidirectional nature of the communication allows the cycle time to be limited only by the speed of the technology. Standard ECL circuitry available in 1992 allows a 2 ns clock. To compare this against a conventional bus, we developed a simple M/G/1 bus model. The model assumes no overhead for arbitration, and single-cycle synchronous transmission in 32-bit chunks. The pin-out for an SCI interface is also 32 bits (16-bit input link plus 16-bit output link).

Figure 9 compares the throughput-latency characteristics of an SCI ring to a bus as the bus cycle time is varied. Data for the SCI ring are from the simulator with flow control in effect. We assume a workload of 60% address packets (16 bytes) and 40% data packets (80 bytes). Ring/bus sizes of 4 and 16 nodes are used. If a synchronous bus had the same cycle time as the SCI ring, it would clearly provide better performance. This is due not only to the bus' greater width, but to the single cycle broadcast latency. With a bus cycle time of 4ns, latency is still lower when lightly loaded, but the maximum throughput is also lower. This is due to the bus' lack of concurrency.

As the bus cycle time is increased, the latency goes up significantly, and the maximum throughput drops off significantly. Realistic bus cycle times range from 20 to 100 ns. A typical high performance shared bus has a 30 ns cycle time. The Stardent Titan graphics supercomputer uses a 31.25 ns bus [Siew91], for example, and the Silicon Graphics Power Series computers use a 30 ns bus [Grap89]. The ELXSI System 6400 used expensive twisted-pair ECL with differential signaling for their shared backplane, achieving a cycle time of 25 ns [Olso83]. The SCI ring provides far greater bandwidth and lower latency than a bus of comparable width running at 20 ns or slower.

As the number of nodes on a ring increases, the average message latency will increase. As the number of nodes on a bus increases, the average message latency will also increase, due to greater contention for the bus and because the cycle time of the bus will have to be increased to accommodate the greater capacitive loading and longer physical distances. Because of the increased cycle time, the total bandwidth of the bus will decrease as well. The cycle time of an SCI ring is independent of ring size.

4.5. Sustained Data Throughput using a Request/Response Model

This section considers total sustained data transfer rates on a ring. We assume that the ring traffic consists solely of read request packets and their associated read response packets. Latencies represent an address packet transmission from a processor to a memory, followed by a data packet transmission from the memory to the processor including receipt of the entire data block (memory

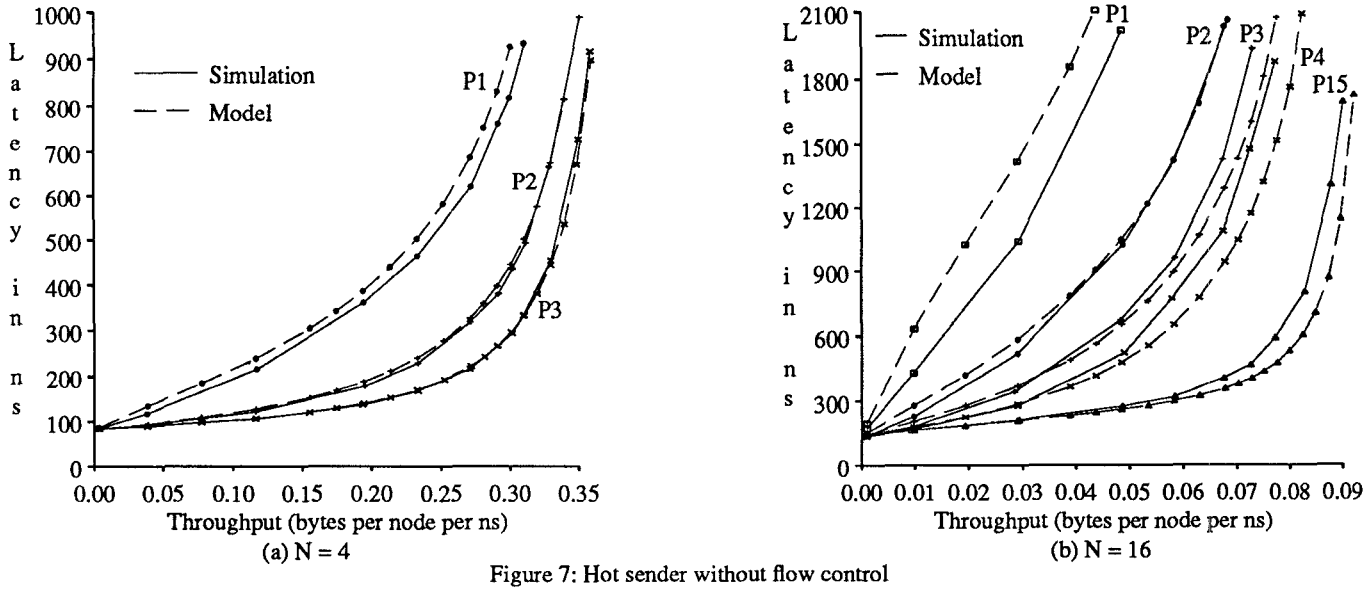


Figure 7: Hot sender without flow control

lookup time is not included). We use a data block size of 64 bytes, and the throughput includes only the data bytes.

The results are shown in Figure 10. Since an address packet is 16 bytes and a data packet includes a 16 byte header along with the 64 bytes of data, exactly two thirds of the send packet symbols contain data. The actual data throughput is thus two thirds of the total throughput. The throughput shown in Figure 10 is the *total* ring throughput, measured in gigabytes per second.

4.6. Breakdown of Message Latency

In this section, we break the mean message latency into several components. Figure 11 plots results from the analytical model for ring sizes of 4 and 16. The packet traffic is uniform, with 40% of the packets containing 64-byte data blocks. The latency is broken into 4 components. The *Fixed* curve represents latency due to wire transmission delay and fixed switching overheads. The *Transit* curve represents the time from when a transmit queue begins transmitting until the packet is consumed at the destination. The difference between these two curves is due to delays passing through the ring buffers. The *Idle Source* curve represents the latency seen by a packet arriving at an idle transmit queue (there are no packets in front of it and the node is not in the recovery stage). The difference between the *Transit* curve and this curve represents the time a source packet may have to wait while a packet finishes passing through the node. The *Total* curve represents total, end-to-end latency. The gap between the *Transit* curve and this curve represents the total time a packet is queued at a transmit queue before receiving permission to transmit.

Most of the latency under heavy loads is due to waiting in the transmit queues. In a closed system (where there is a limit on the number of queued packets), the delay due to transmit queueing would level off at some point. Delay due to buffer backlog becomes more significant relative to transmit queueing delay as the ring size is increased from 4 to 16. For very large rings, transmission delay becomes dominant except when the ring is very close to saturation.

4.7. Discussion of Model

Below we identify several possible sources of error in our analytical model, and discuss the likely significance of each.

First, we assume geometrically distributed inter-packet-train spacing, whereas simulations show that certain lengths of spaces are much more common. For example, the space created

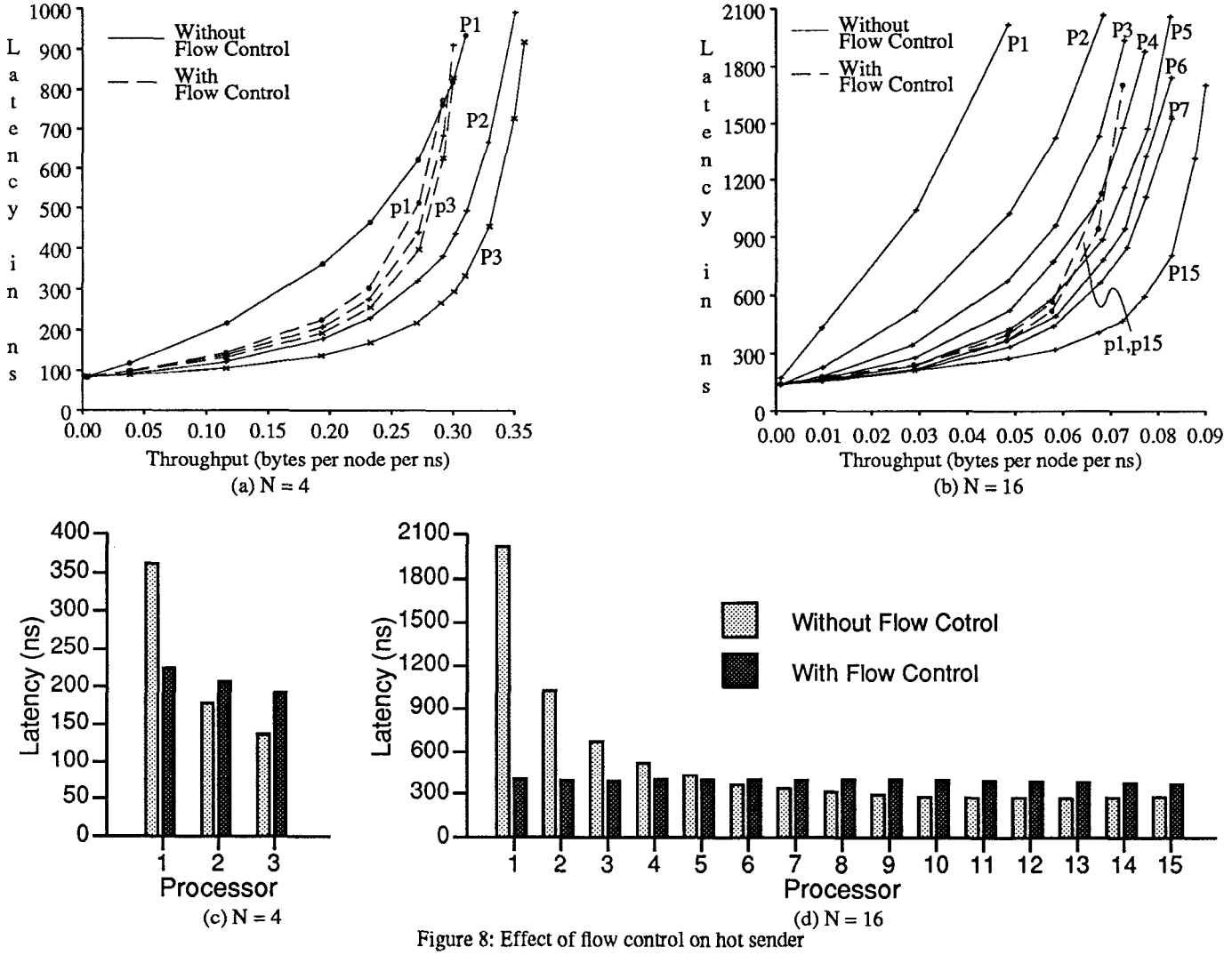


Figure 8: Effect of flow control on hot sender

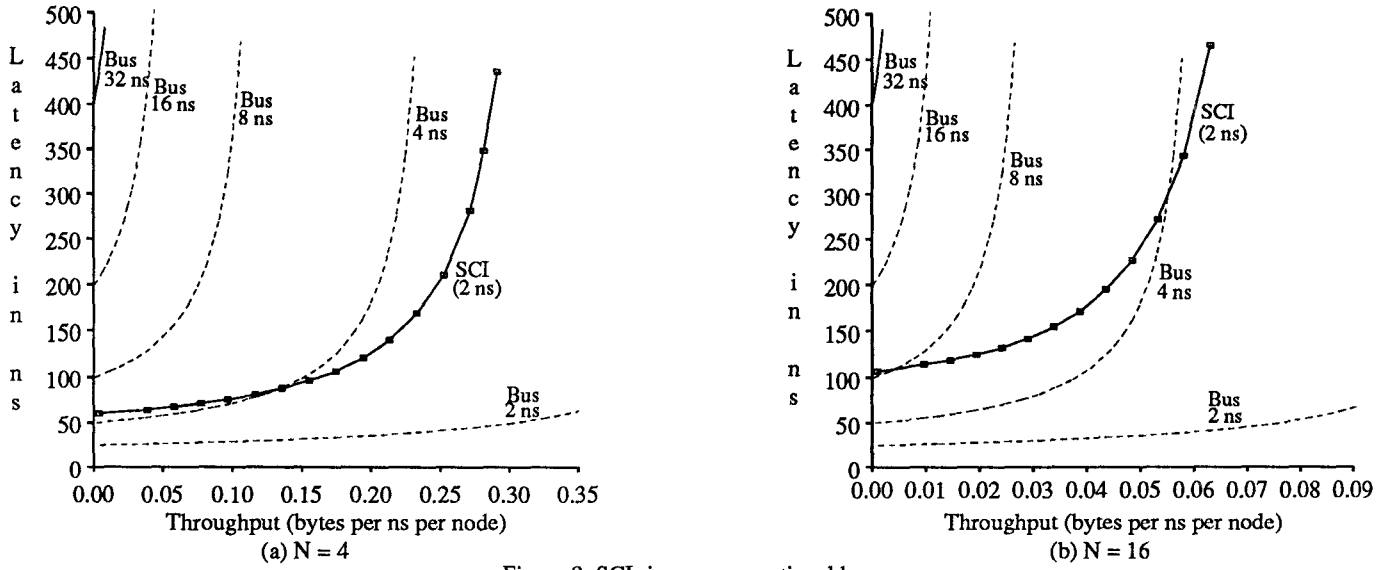


Figure 9: SCI ring vs conventional bus

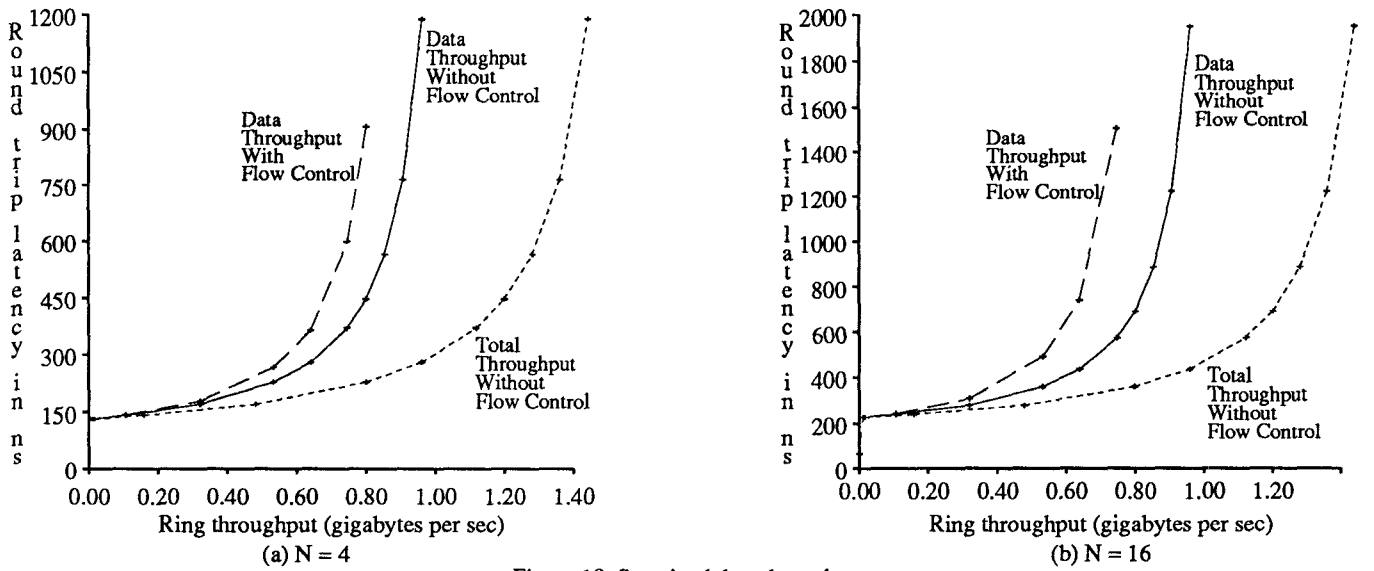


Figure 10: Sustained data throughput

by replacing an address packet with an echo packet occurs with high frequency. However, simulation estimates of the coefficient of variation of the inter-packet-train spacing, are very close to 1. Thus, we do not anticipate that this assumption causes significant error.

Second, in computing the variance of the recovery period, we do not know the correlation between the residual life of a passing packet train and the remaining portion. We assume a correlation of one, and treat the portion due to residual life of a passing packet train as a constant multiplier to the remaining recovery time (Equation (25)). This slightly overestimates the service time variance, but we do not believe this causes significant error.

Finally, we assume that the transmit queue utilization and the pass-through ring utilization are independent. That is, the model assumes we see the same rate of passing packets regardless of whether the transmit queue is in use. *This is the primary source of error in the model.* We have found using the simulator that the pass-through traffic tends to be lower than average when the transmit queue is idle, and higher than average when the transmit queue is active (during the transmission/recovery stage). This causes the model to underestimate the length of the recovery stage, thus underestimating the overall message latency. The error increases as the mean length of the recovery period increases, which causes the error to grow for larger rings and packet sizes.

Two worthwhile directions for future research are to reduce the error in the current model and to extend the model to account for flow control.

5. CONCLUSIONS

This paper presented a performance study of the SCI ring, including a description of the logical-level protocol, an efficient, analytical performance model, and extensive simulation results. The performance of the SCI ring was analyzed under uniform and non-uniform workloads and with and without the flow control mechanism. The SCI ring was also compared to a conventional shared bus.

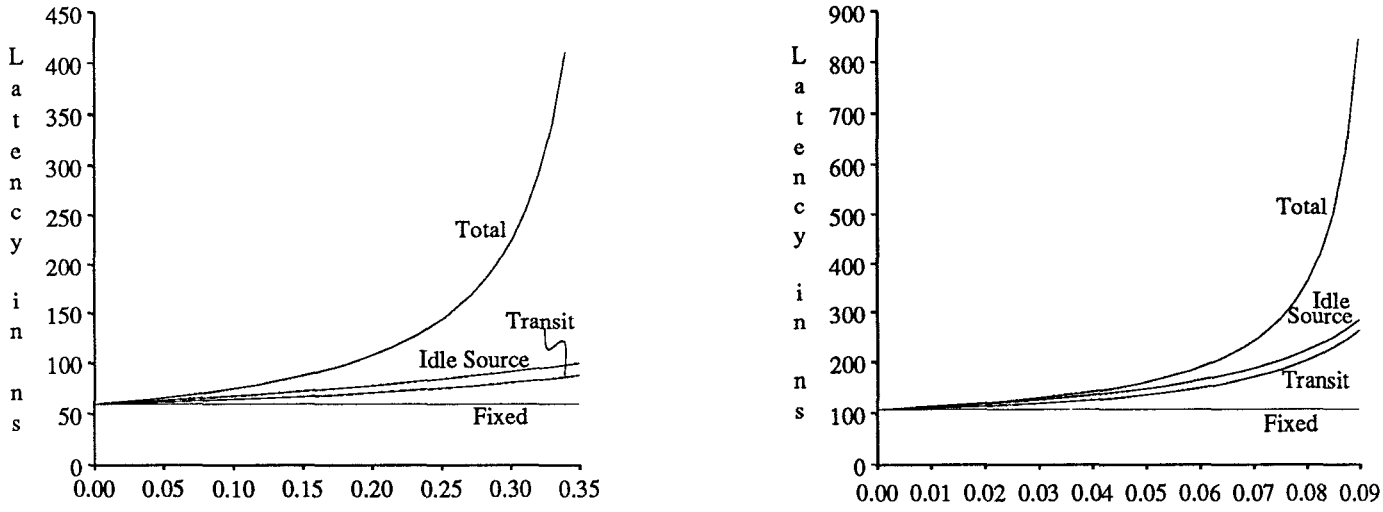


Figure 11: Breakdown of Message Latency

The analytical model developed for the SCI ring did not consider the flow control mechanism, but was found to be accurate for both uniform and non-uniform communication patterns. Where quantitative error was greater, qualitative behavior was still predicted correctly. The primary source of error in the model was identified, and will be the topic of further research, along with extensions to model flow control mechanisms.

The SCI flow control mechanism effectively prevents node starvation, providing all nodes with their approximate fair share of the ring bandwidth. Non-uniform routing still affects the realized node throughputs to some extent, however, with the effect being greater for smaller ring sizes. The flow control mechanism also equalizes the negative impact that a hot node has on the rest of the ring. Without flow control, the downstream neighbors of a hot node see substantially increased message latencies.

The fairness provided by the flow control mechanism comes at the cost of overall ring throughput. Maximum throughput is reduced by up to 30%. The impact is greatest for ring sizes of 8 to 32, and is negligible for a ring size of 2. We are investigating possible modifications to the flow control mechanism that would gracefully increase ring throughput in return for reduced fairness.

In comparing the SCI ring to a conventional bus, the clock speed was considered along with the number of cycles needed to convey a message. Although the number of cycles is larger for the ring, the faster clock speed gives a significant advantage. A 32-bit bus would have to have a 4 ns clock to be competitive with a 16-bit wide SCI ring with a 2 ns clock (and even then it would have a lower saturation bandwidth). While a 2 ns clock for SCI is realizable in 1992 with standard ECL circuitry, typical high performance multiprocessor buses have cycle times of about 30 ns.

With a 16-bit width and 2 ns cycle time, the SCI ring provides a total peak throughput of over 1 gigabyte per second. Communication locality and larger packet sizes increase this number, while flow control overhead decreases it. Given a read request/read response model and 64-byte data blocks, a total data transfer rate of approximately 600-800 megabytes per second can be sustained over a single ring. The flow control mechanism partitions this bandwidth fairly among all participating nodes. The SCI standard leaves room for future improvements by both increasing the link width and decreasing the cycle time.

Acknowledgements

The authors gratefully acknowledge the assistance of Dave James, Ross Johnson and Alain Kagi in clarifying several important concepts in the SCI standard. Thanks also to Rich Maclin and Ross Johnson for providing valuable mathematical assistance. In addition we have received much help from other SCI colleagues too numerous to mention, but especially from Dave Gustavson, Stein Gjessing, Ernst Kristiansen, and Hans Wiggers.

Steven Scott was supported by a fellowship from the Fannie and John Hertz Foundation. This work was also supported by National Science Foundation grants #CCR-892766 and #CCR-9024144.

References

[Grap89] Silicon Graphics Inc., Power Series, Silicon Graphics Technical Report, 1989.

[Gust92] Gustavson, D. B., The Scalable Coherent Interface and Related Standards Projects, *IEEE Micro* **12**(1), February 1992, 10-22.

[IEEE91] IEEE, SCI-Scalable Coherent Interface, P1596/D2.00 18Nov91, Draft for Recirculation to the Balloting Body, prepared by the P1596 Ballot Review Committee of the IEEE Microprocessor Standards Committee, 1991.

[Klei75] Kleinrock, L., *Queueing Systems, Volume I,* John Wiley and Sons, New York, 1975.

[Olso83] Olson, R. A., B. Kumar, and L. E. Shar, Messages and Multiprocessing in the ELXSI System 6400, *COMPCON83*, February 1983, 21-24.

[Scot91] Scott, S. L., J. R. Goodman, and M. K. Vernon, Analysis of the SCI Ring, Computer Sciences Technical Report #1055, University of Wisconsin-Madison, Madison, WI 53706, November 1991.

[Siew91] Siewiorek, D. P. and P. J. Koopman, Jr., *The Architecture of Supercomputers: Titan, A Case Study,* Academic Press, San Diego, CA, 1991.

Appendix A — Performance Model Equations

Model inputs:

N Number of nodes per ring
$z_{i,j}$ Fraction of node i's packets routed to node j
λ_i Transmit queue packet arrival rate at node i
f_{data} Fraction of send packets that are data packets
f_{addr} Fraction of send packets that are addr packets
l_{data} Length (in symbols) of a data packet (including post-pended idle)
l_{addr} Length of an address packet
l_{echo} Length of an echo packet
T_{wire} Number of cycles to traverse a wire
T_{parse} Number of cycles to parse a packet

Preliminary calculations:

l_{send} Mean length of a send packet

$$l_{send} = f_{data}\, l_{data} + f_{addr}\, l_{addr} \tag{1}$$

X_i Mean throughput at node i

$$X_i = \lambda_i\,(l_{send}-1) \tag{2}$$

λ_{ring} Total packet arrival rate

$$\lambda_{ring} = \sum_{i=0}^{N-1} \lambda_i \tag{3}$$

$r_{echo,i}$ Rate of echo packets passing through node i

$$r_{echo,i} = \sum_{j\neq i} \lambda_j \sum_{\substack{k=j+1 \\ (mod\,N)}}^{i} z_{jk} \tag{4}$$

$r_{data,i}$ Rate of data packets passing through node i

$$r_{data,i} = f_{data} \sum_{j\neq i} \lambda_j \sum_{\substack{k=i+1 \\ (mod\,N)}}^{j-1} z_{jk} \tag{5}$$

$r_{addr,i}$ Rate of address packets passing through node i

$$r_{addr,i} = f_{addr} \sum_{j\neq i} \lambda_j \sum_{\substack{k=i+1 \\ (mod\,N)}}^{j-1} z_{jk} \tag{6}$$

$r_{pass,i}$ Total rate of packets passing through node i

$$r_{pass,i} = r_{echo,i} + r_{data,i} + r_{addr,i} = \sum_{j\neq i} \lambda_j \tag{7}$$

$r_{rcv,i}$ Rate of packets routed to node i

$$r_{rcv,i} = \sum_{j\neq i} \lambda_j z_{ji} \tag{8}$$

$n_{pass,i}$ Mean number of passed packets per injected packet at node i

$$n_{pass,i} = \frac{r_{pass,i}}{\lambda_i} \tag{9}$$

$U_{pass,i}$ Utilization of node i's output link by passing packets

$$U_{pass,i} = r_{data,i} l_{data} + r_{addr,i} l_{addr} + r_{echo,i} l_{echo} \tag{10}$$

$l_{pkt,i}$ Mean length of a passing packet at node i

$$l_{pkt,i} = \frac{U_{pass,i}}{r_{pass,i}} \tag{11}$$

$L_{pkt,i}$ Residual life of passing packet at node i

$$L_{pkt,i} = \left[\frac{r_{data,i} l_{data}^2 + r_{addr,i} l_{addr}^2 + r_{echo,i} l_{echo}^2}{2U_{pass,i}}\right] - \frac{1}{2} \tag{12}$$

Calculations inside iteration:

$n_{train,i}$ Mean number of packets in a passing packet train at node i

$$n_{train,i} = \frac{1}{1-C_{pass,i}} \tag{13}$$

$l_{train,i}$ Mean length of passing packet train at node i

$$l_{train,i} = l_{pkt,i} n_{train,i} \tag{14}$$

$P_{pkt,i}$ Probability that an idle symbol passing through node i is directly followed by a packet

$$P_{pkt,i} = \frac{U_{pass,i}}{(1-U_{pass,i}) l_{train,i}} \tag{15}$$

S_i Mean transmit queue service time at node i

$$S_i = (1-\rho_i) U_{pass,i} [L_{pkt,i} + (C_{pass,i} - P_{pkt,i}) l_{train,i}] + l_{send}\,(1+P_{pkt,i} l_{train,i}) \tag{16}$$

ρ_i Utilization of transmit queue at node i

$$\rho_i = \lambda_i S_i \tag{17}$$

Calculating new coupling probabilities:

$C_{link,i}$ Probability that packet on node i's output link immediately follows its predecessor

$$C_{link,i} = \frac{1}{(n_{pass}+1)} \left[n_{pass,i} C_{pass,i} + [\rho_i + (1-\rho_i) U_{pass,i}] + P_{pkt,i} l_{send} \right] \tag{18}$$

$F_{in,i}$ Mean number of "following" packets entering stripper i per stripped packet

$$F_{in,i} = C_{link,i-1} \left[\frac{\lambda_{ring}}{\lambda_i + r_{rcv,i}}\right] \tag{19}$$

$P_{uncouple}$ Probability a stripped packet at node i causes the following packet to become uncoupled, given there is a following packet

$$P_{uncouple,i} = \left[\frac{\lambda_i}{\lambda_i + r_{rcv,i}}\right] \left[\frac{\lambda_{ring} - \lambda_i - r_{rcv,i}}{\lambda_{ring}}\right] \tag{20}$$

F_{out} Mean number of "following" packets leaving stripper i per stripped packet

$$F_{out,i} = (1-C_{link,i-1})^2 F_{in,i} + C_{link,i-1}(1-C_{link,i-1})(F_{in,i}-1) + C_{link,i-1}^2 (F_{in,i} - 1 - P_{uncouple,i}) + (1-C_{link,i-1}) C_{link,i-1} (F_{in,i} - P_{uncouple,i}) \tag{21}$$

$C_{pass,i}$ Probability that passing packet at node i immediately follows its predecessor

$$C_{pass,i} = \frac{F_{out,i}(\lambda_i + r_{rcv,i})}{\lambda_{ring} - \lambda_i} \tag{22}$$

Calculating final model outputs:

$V_{pkt,i}$ Variance of passing packet length at node i

$$V_{pkt,i} = \frac{1}{r_{pass,i}}\Big[r_{data,i}(l_{data}-l_{pkt,i})^2 + r_{addr,i}(l_{addr}-l_{pkt,i})^2 + r_{echo,i}(l_{echo}-l_{pkt,i})^2\Big] \quad (23)$$

$V_{train,i}$ Variance of passing packet train length at node i

$$V_{train,i} = \frac{V_{pkt,i}}{(1-C_{pass,i})} + \frac{l_{pkt,i}{}^2 C_{pass,i}}{(1-C_{pass,i})^2} \quad (24)$$

$\psi_{type,i}$ Constant multiplier for approximate variance calculation

$$\psi_{type,i} = \frac{(1-\rho_i)U_{pass,i}[L_{pkt,i} + (C_{pass,i}-P_{pkt,i})l_{train,i}] + l_{type}P_{pkt,i}l_{train,i}}{l_{type}P_{pkt,i}l_{train,i}} \quad (25)$$

where $type \in \{addr, data\}$

$V_{addr,i}$ Variance of service time for an address packet injected at node i

$V_{data,i}$ Variance of service time for a data packet injected at node i

$$V_{type,i} = \left[\sum_{j=1}^{l_{pkt}}\binom{l_{type}}{j}P_{pkt,i}{}^j(1-P_{pkt,i})^{l_{pkt}-j}(jV_{train,i} + [jl_{train,i}]^2) - (l_{train,i}P_{pkt,i}l_{type})^2\right]\psi_{type,i}^2 \quad (26)$$

V_i Overall variance of service time for an packet injected at node i

$$V_i = f_{data}(V_{data,i} + S_{data,i}{}^2) + f_{addr}(V_{addr,i} + S_{addr,i}{}^2) - S_i{}^2 \quad (27)$$

c_i Coefficient of variation of S_i

$$c_i = \frac{\sqrt{V_i}}{S_i} \quad (28)$$

Q_i Mean transmit queue length at node i

$$Q_i = \rho_i + \frac{\rho_i{}^2(1+c_i{}^2)}{2(1-\rho_i)} \quad (29)$$

L_i Mean residual life of transmit queue service time at node i

$$L_i = \frac{V_i + S_i{}^2}{2S_i} \quad (30)$$

W_i Mean wait time in transmit queue at node i

$$W_i = (Q_i - \rho_i)S_i + \rho_i L_i \quad (31)$$

B_i Mean backlog at node i seen by a passing packet

$$B_i = \Bigg[(1-\rho_i)U_{pass,i}(C_{pass,i}-P_{pkti})l_{send}n_{train,i} + f_{data}P_{pkt,i}l_{data}\left[\frac{l_{data}+1}{2}\right]n_{train,i} + f_{addr}P_{pkt,i}l_{addr}\left[\frac{l_{addr}+1}{2}\right]n_{train,i}\Bigg]\frac{1}{n_{pass,i}} \quad (32)$$

T_i Mean transit time for node i (once transmission begins)

$$T_i = 1 + T_{wire} + t_{parse} + l_{send} + \sum_{j\neq i} z_{ij}\sum_{\substack{k=i+1 \\ (mod\ N)}}^{j-1}(1 + T_{wire} + t_{parse} + B_k) \quad (33)$$

R_i Mean response time of a packet transmission from node i

$$R_i = W_i + (1-\rho_i)U_{pass,i}L_{pkt,i} + T_i \quad (34)$$