Measures of Message Latency for Wormhole Routed Networks

Kant C. Patel and D. N. Jayasimha[1]

Department of Computer and Information Science

The Ohio State University

Columbus, Ohio 43210–1277

Email: {patel, jayasim}@cis.ohio-state.edu

Phone: (614) 292–6653

Fax: (614) 292–2911

[1]Part of this work was done while the author was on leave at NASA Lewis Research Center, Cleveland, OH during 1993-94.

Abstract

A traditional measure of the communication performance in a massively parallel processor (MPP) is the average message latency. In this paper we argue that the variance, or equivalently, the standard deviation of message latencies should be considered as an additional important measure in a multi-user MPP environment. Using a simulation based approach, we investigate various schemes to reduce the standard deviation without adversely affecting the average latency. Simulations on a 16×16 wormhole routed mesh architecture reveal that the standard deviation varies typically from 25% to 50% of the average depending on the message generation rate, the message length, the traffic pattern, and the routing algorithm. Since the differential blocking faced by messages is one of the major causes for the large standard deviation, we take two approaches to reducing it which do not degrade the average. The first controls the relative progress with which different messages proceed towards their destinations by *prioritizing* them dynamically. The second employs multiple virtual channels per physical channel. We compare the performance of several schemes for each of these two approaches, using deterministic routing as our base case. A distinguishing feature of our investigation is that the simulation model accounts for the degradation in network cycle time with increased router complexity required by the use of the priority schemes and by the virtual channels. In fact, we show how our experimental results could have been misleading had we followed the traditional "cycle counting" approach.

Keywords: wormhole routing, message priorities, virtual channels, router design complexity, average message latency, standard deviation, performance, simulation.

1 Introduction

Minimizing message latency is of primary interest in massively parallel processors (MPPs). Traditionally, the average message latency and the network utilization have been used as performance measures to evaluate and propose routing algorithms. The application developer would like to see as low a message latency as possible, for it would be a fairly good indicator of the overall performance of the application. On the other hand, the system administrator would like to see as high a network utilization as possible, for it would be a fairly good measure of the machine performance as a whole. So, the primary concern while designing the various routing algorithms has been to reduce the average network latency as well as to increase the average network utilization. In this paper, we look at the problem of large variance, or equivalently, standard deviation of message latencies for interprocessor communication in wormhole-routed interconnection networks. Our initial focus is on 2-D mesh architectures although our study can be extended to k-ary n-cubes. It should be noted that wormhole-routed, k-ary n-cube networks have become the interconnection pattern of choice in most modern, distributed-memory multiprocessors such as the Cray T3D, the MIT J-machine, the Caltech MOSAIC, the Intel Paragon, and the nCUBE-2.

The paper is organized as follows: Section 2 explains why it is worthwhile to look at the standard deviation of message latencies and describes previous work done in closely related areas. Section 3 describes the methodology used in our study. Different schemes attempting to reduce the standard deviation of latencies have been described in Sections 4 and 5. Section 6 describes the methodology used to evaluate these schemes, and simulation results are presented in Section 7. Section 8 summarizes the work and presents other ongoing work in this direction.

2 Background and Motivation

In this paper we argue that the standard deviation of message latencies should be considered as an important latency measure in addition to the average latency measure in multi-user, multiprogrammed MPP environments and in real-time parallel processing applications. Traditionally, most researchers have concentrated on reducing the average latency measure or on increasing the network throughput. Dally and Seitz [3, 4] have proposed the use of multiple virtual channels primarily to reduce the average message latency and to increase the network throughput. Some schemes based on different input selection policies have been proposed by Glass and Ni [8], J. Duato [6], S. Konstantinidou [10], to alleviate the effect of network congestion on message latency.

For cost-effectiveness and better throughput, MPPs need to support multiple users at the same time — many multiprocessors such as the CM-5 and the Intel Paragon do. When a random mix of applications is executing on a multiprocessor, the average message latency for the entire network is no longer the only concern. The average latencies faced by two applications might vary drastically (even though the *overall* average latency remains the same). This would be unfair to some users. Further, different runs of the same application might face different execution times with the same network load because of varying message latencies. It is therefore important and desirable that the variance of message latencies be kept low.

In real-time parallel processing, worst case latencies are a major factor in determining how data is accessed, buffered, and transmitted in a predictable manner [12]. It is important, therefore, that for good performance, the worst case latency be as close to the average as possible. Equivalently, it is important to reduce the variance of messages in such applications.

To our knowledge, the issue of variance of latencies has been addressed in detail only recently by Mraz [12]. Dally mentions briefly that the use of a particular scheduling strategy, called oldest packet first, possible with the use of virtual channels, does reduce the variance in message latency [2]. Mraz's work examines the variance that can arise in the message latency when high priority operating system functions and interrupts preempt the execution of *send* and *receive* primitives. The nature of Mraz's experiments does not lead to message contention in the network and hence the latter does not contribute to variance in message latencies.

Our study complements Mraz's by considering the effect of network contention on the average and the variance of message latencies. The network contention depends, for example, on the network load, the message traffic pattern, the routing algorithm, the input and output selection policies at a router, the mapping of processes to processors, etc.

Since the standard deviation, which is the square root of the variance, is the usual measure of dispersion, we will use the former instead of the latter measure throughout the paper. We are interested then in a) a low standard deviation of messages that travel the same distances *and* b) a low or reasonable standard deviation of messages which travel different distances. The latter is important if message latencies are to be largely independent of distance and in cases where the performance of the algorithm is not sensitive to "bad" mappings which is likely to be the case in a multi-user environment. In this paper, our focus is on the standard deviation and the average message latency considering all message lengths. We do make some observations, however, regarding the standard deviation of messages traveling the same distance also.

3 Overview

After looking at different possible approaches for reducing the standard deviation without degrading the average message latency, we decided to investigate the use of a) message priorities (Section 4), and b) multiple virtual channels and adaptive routing algorithms (Section 5). The reasons for selecting these approaches for study are presented in the following sections, along with a description of the different schemes that we used for each approach. Our simulation-based study takes into account the degradation in the network cycle time that arises by using additional resources at a router. The architecture that we have selected for our study is a wormhole routed 2-D mesh. In general, for each scheme, we first look at its effect on the average and the standard deviation, *without changing the network cycle time*. We compare the performance of this scheme with the usual X-Y routing algorithm, our basis for comparison. If there is little or no improvement or a degradation in the average and the standard deviation, then we conclude that that particular scheme will not work well in practice. On the other hand, if a moderate to remarkable improvement is observed, then we reevaluate the scheme by considering the degradation in network cycle time as a result of the increased router complexity and recommend this scheme only if it performs well under this scenario.

Using this methodology has led to some surprising results with each approach, especially, with the use of multiple virtual channels and adaptivity. These results are presented in Section 7. We point out in that section that, when comparing routing algorithms, the traditional method of measuring latency by counting the number of cycles could lead to inaccurate and possibly misleading results.

4 Prioritizing Messages

4.1 Motivation

When several messages contend for an outgoing channel at a router, they are queued up and get the channel in a first-in-first-out fashion, irrespective of the blocking suffered till then. For example, the header flit of a message generated locally (at that processor) might be in front of other headers (belonging to messages generated at other processors) in the queue, but it might have faced very little delay so far, whereas the other messages might have faced substantial delays at each node that they have passed through. However, the basic scheme will allow the locally generated message to go first in this case, as it is at the head of the queue, thereby further delaying the messages which have already faced substantial delay. Owing to this, messages traveling the same distance could face widely varying delays. Since this differential blocking could be one of the contributors to a large standard deviation, we associate a priority with each message, which changes dynamically as the message is routed through the network. Thus by modifying the priority of a message, the relative progress with which it proceeds towards its destination could be controlled.

4.2 Different Schemes Used

We have considered several priority schemes, two of which, closest-first [14] and distance-traveled [8, 7] are based only on some measure of distance. In the closest-first scheme, the priority of a message is directly proportional to the inverse of distance between the current node and its destination, and in the distance-traveled scheme, the priority of a message is directly proportional to the distance that it has traveled so far. Theoretically, any scheme that is based solely on distance is bound to be prone to indefinite postponement, thereby possibly producing a negative effect on the standard deviation of message latencies. Hence, we propose schemes based on some function of distance and blocking delay– blocking delay is defined as the cumulative delay faced by the message owing to its blocking at each node on its path so far. This blocking delay does not include the delay owing to the header processing time and the flit transfer time. Rather, it is the delay faced by a message while it is blocked, waiting for an output channel to become available. The use of blocking delay does not require a global clock, as it is an accumulation of the delays faced locally by the message header while it is blocked at the intermediate nodes– it can be maintained and updated by computing the local delays at each node on the path of the message as it gets routed. If a message does not get blocked at any intermediate node, then its blocking delay is zero. The exact blocking delay is expensive to compute since it would involve an addition (possibly a floating point addition) and additional hardware. We have also considered several approaches to approximate the blocking delay by using counters and shift-registers, thereby eliminating the use of expensive hardware and replacing floating-point computations by increments and shifts on integers. Using these approaches, multiplications can also been approximated by binary shifts using a barrel shifter. A detailed discussion of these approaches is outside the scope of this paper.

4.2.1 Blocking Delay

In this scheme, the priority of a message is directly proportional to the value of blocking delay associated with it. The idea is to implement some kind of fairness by giving higher priority to messages that have faced higher blocking delays. The standard deviation of latencies is expected to decrease because the messages that have blocked more are more likely to proceed towards their destination at the cost of those that have blocked lesser.

4.2.2 Closest-First with ties broken by Blocking Delay

This is similar to the closest first scheme proposed in [14], but now we use the blocking delay to break ties, i.e., if several messages contending for an outgoing channel at a node have the same distance left to be traveled, the one with the largest value of blocking delay associated with it is allowed to reserve that channel.

4.2.3 Normalized Blocking Delay

In this scheme, the blocking delay is normalized with respect to the total path length. At moderate to high traffic, the messages traveling a shorter distance have lower average latency compared with those traveling a longer distance. Further, shorter messages need to reserve a smaller number of channels and hence are given priority. This scheme could have an adverse effect on the latency measures of messages needing to travel longer distances, however.

4.2.4 Blocking Delay $\times$ Path Length

A message traveling distance X will reserve more channels than another one traveling distance Y, where X $>$ Y, and hence, will block more messages. So, we try to reduce blocking by giving higher priority to message X– thus, the priority of a message is made proportional to the product of blocking delay and path length.

The use of unweighted path lengths, as in the previous schemes, unduly favors messages that have to travel longer distances at the cost of those that travel shorter distances. We observe that under uniform traffic, maximum number of messages have a path length of *diameter/3*, where *diameter* is the maximum distance between any two nodes in the 2-D mesh, and the number of messages decreases as we move away from this path length, on either side. Hence, the weight assigned to path length l is proportional to the frequency at which messages traversing a path length of l are generated. The priority, then, is directly proportional to the product of blocking delay and the weighted path length. This scheme, however, did not perform as well as the others described previously. Hence, we have not reported these results.

5 Use of Multiple Virtual Channels and Adaptivity

5.1 Motivation

The use of multiple virtual channels per physical channel should reduce the blocking of messages at each intermediate node on their paths, thereby reducing channel contention. As a result, more messages are able to concurrently proceed towards their destinations than the case when fewer or no virtual channels are used. So, owing to an overall reduction in message blocking, this approach is expected to reduce the standard deviation. Furthermore, at high traffic, the use of virtual channels reduces the average message latency too ([4, 5, 2, 6]). Since the channel bandwidth is shared among more than one message, the average message latency is expected to increase when the network traffic is low, however. We assume that the virtual channels are time-multiplexed.

5.2 Different Schemes Used

We vary the number of virtual channels per physical channel for deterministic (X-Y) routing. The routing function still remains the same as for basic X-Y routing in the sense that it decides, in a deterministic manner, whether the header should routed along X+/X- or Y+/Y- directions. The only modification is that for each routing decision, there are multiple virtual lanes (channels) to choose from, rather than just one as in the base case. For example, if two virtual channels are used per physical channel, then a decision to route a message along X+ direction at a particular router means that the message can choose either of the two channels going along X+ out of that router. The effect of adaptivity on average and standard deviation of latencies has been studied by using partially adaptive (for example, West-first [9]), and fully adaptive (for example, Opt-y [13]) routing algorithms. West-first has roughly half the adaptiveness of a fully adaptive routing algorithm. Opt-y has been proved to be optimal in the number of virtual channels required and in the number of restrictions placed on the use of these virtual channels [13].

6 Simulation Setup

Simulation has been used to evaluate the different approaches described in previous sections. We have varied the message length, message generation frequency, traffic pattern and routing algorithm to see how well the different approaches work with different values of these parameters on a 2D-mesh topology.

We use a CSIM-based simulation testbed to evaluate the different approaches described in previous sections. A distinguishing feature of our investigation is that the simulation model accounts for the degradation in network cycle time with increased router complexity required by the use of the priority schemes and by the virtual channels. This is based on the cost model proposed by Chien [1][1]. The simulator does not incorporate the effect of message start-up times explicitly, but for most of the network traffics used for our experiments, the average interval between generation of consecutive messages from each node is larger than typical values of start-up times.

We have evaluated the different schemes on a 16×16 mesh. Messages having lengths of 16, 32, 64, 128, 160 and 192 flits have been considered. Uniform traffic and hotspot traffic patterns have been used. In uniform traffic, each node generates a message with a given probability, per cycle, for a randomly chosen destination. In hotspot traffic, each node generates a message with a given probability, per cycle. The destination of the message is chosen to be the node at the center of the quadrant of the mesh containing the source node, with probability 0.40, and is chosen randomly with probability 0.60. We assume, as other researchers do, that the uniform and hotspot traffic patterns are representative ones for a variety of workloads. It should be noted, however, that any synthetic traffic patterns considered for simulations will only give an approximation to the real situation of experimenting with actual workloads on a real machine.

As mentioned earlier, the cost model proposed by A.Chien [1] has been used to determine the values for router setup delay, which is the time to create a connection through a router, and the flow control cycle time, which is the time to do a flow control operation. This model has been instantiated for the 0.8 micron CMOS gate array technology. We have shown below the computation of setup delay and flow control cycle time, for dimension-order routing with a single physical channel, as done in [1].

In case of dimension-order routing, dimensions are changed in order. Hence, the router requires only one crossbar per dimension. The inputs to the crossbar in the X-dimension are from the processor, and $X+$ and $X-$ directions. The outputs are to the Y-dimension, and the $X+$, $X-$ directions.

[1]We also considered some improvements to Chien's model, such as reducing the crossbar complexity, using three input logic gates, etc. which reduce the cycle times. However, the ratio of the cycle times for routing with virtual channels to that without virtual channels does not change significantly owing to these modifications. Hence, the cycle times used for our simulations are based on [1].

Similarly, a 3×3 crossbar is needed in the Y-dimension. Thus, P=3, F=3 and V=0, where P is the number of inputs or outputs for the crossbar in the router, F is the routing freedom, and V is the number of virtual channels multiplexed onto a physical channel. According to the model, the equation for setup delay can be written as:

$$T_{setup} = T_{AD} + T_{ARB} + T_{CB}$$

where T_{AD} is the time required for decoding address and generating correct requests, T_{ARB} is the time required for arbitration, T_{CB} is the time required to drive data through the crossbar. Hence,

$$T_{startup} = c_3 + c_4 + c_5 * lgF + c_0 + c_1 * lgP$$

where $c_i's$ are constants [1]. This gives a setup delay of 5.6 nanoseconds. Similarly,

$$T_{flow\_control} = T_{FC} + T_{CB}$$

where T_{FC} is the flow controller delay, and T_{CB} is the forward crossbar switch delay for data. Hence,

$$T_{flow\_control} = c_2 + c_0 + c_1 * lgP$$

This gives a flow control cycle time of 3.55 nanoseconds. Hence, the network cycle time is 3.55, the minimum of setup delay and flow control cycle time. Consequently, setup delay is modeled by two network cycles and flow control cycle time by one network cycle.

Similar computations for other cases yield the results summarized in Table 1.

Table 1: Router setup delay and Flow control cycle times

Routing Algorithm	**Network Cycle Time** (nanoseconds)	**Setup Delay** (# network cycles)	**Flow Control** (# network cycles)
Simple X-Y	3.55	2	1
West-first	3.80	3	1
X-Y (2 VCs)	5.83	2	1
Opt-y	6.12	2	1
X-Y (3 VCs)	6.48	2	1

7 Results

In this section, we present the results and explanation for a representative set of these experiments conducted using the simulation setup described in the previous section. In all the graphs, network traffic is represented by a normalized measure, i.e., the number of flits injected into the whole network per nanosecond. The latency measures are normalized with respect to message length and are represented in terms of nanoseconds per flit.

7.1 Latency Measures – The Base Case

We will use the average latency and the standard deviation of latencies for X-Y routing with one physical channel in each direction, and one virtual channel per physical channel, as the basis of comparison with all the other schemes outlined in Sections 4 and 5.

7.1.1 Effect of network traffic

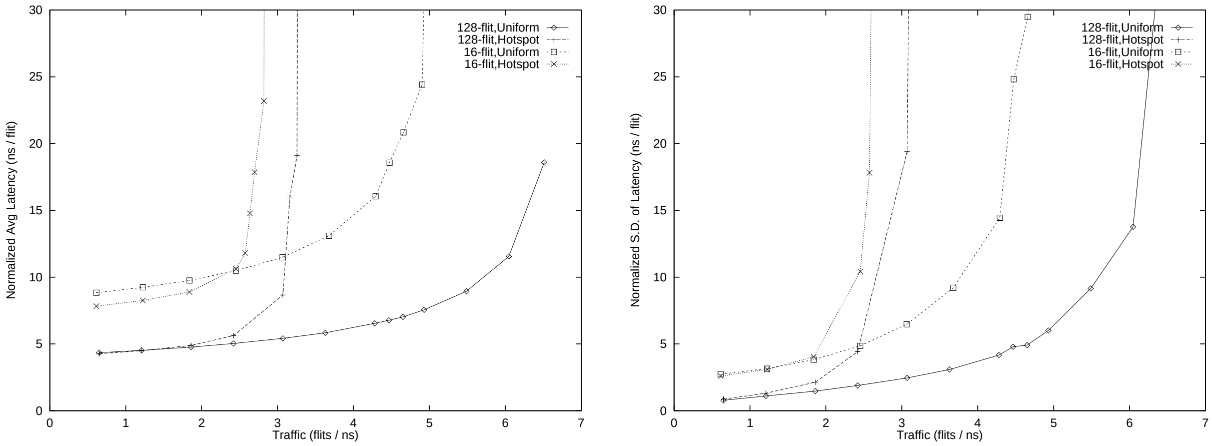


Figure 1: Normalized Latency Measures vs. Traffic for XY routing on a 16x16 mesh with 16-flit and 128-flit messages

With X-Y routing, it was observed that the average channel utilization increased linearly from 2.5% to 25.5% with an increase in network traffic from 0.6 flits/cycle to 6.5 flits/cycle. As can be expected, the average latency, μ, increases almost linearly with increase in network traffic, at low to moderate traffic. The rate of increase is superlinear as the network approaches saturation (see Figure 1). The standard deviation, σ, also increases with increase in network traffic, and for most cases, it is in the range of 25% to 50%. Using *Chebyshev's theorem*, at least 75% of the messages have latencies in the interval [0.5μ, 1.5μ] when σ = 25%, and in the interval [0, 2μ] when σ = 50%, which is clearly very high. Hence, we need some mechanisms to reduce the standard deviation. Further, it can be observed from Figure 1 that the rate of increase of standard deviation with increase in traffic is even higher than that of the average. The average and standard deviation keep on increasing since more and more number of messages get blocked as the traffic increases. The blocked messages make parts of the network unusable which in turn leads to further blocking, finally resulting in the network getting saturated. This blocking behavior is typical of wormhole routing.

7.1.2 Effect of message length

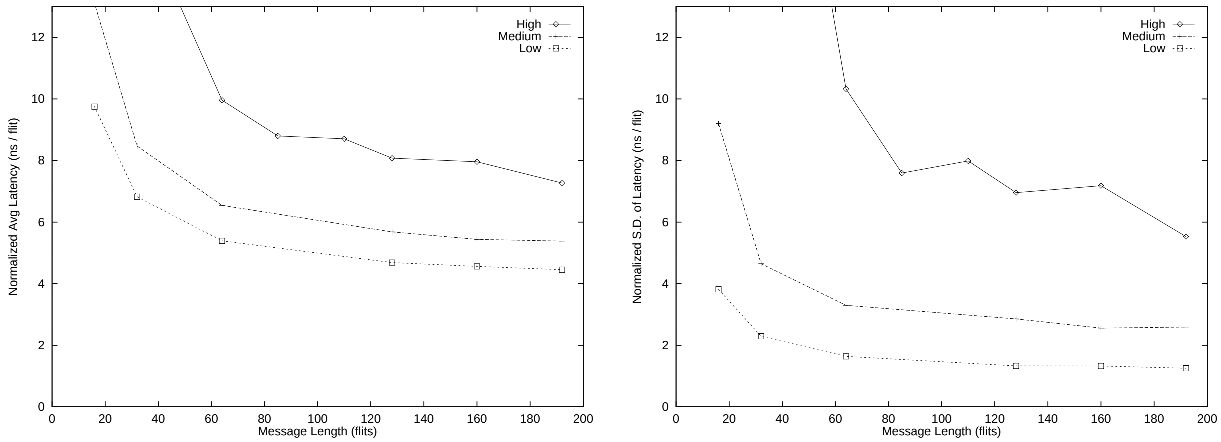


Figure 2: Normalized Latency Measures vs. Msg. Length for XY routing under Uniform traffic on a 16x16 mesh with injection rates, in flits/cycle, of: (High) 19.6, (Medium) 13.0, (Low) 6.5

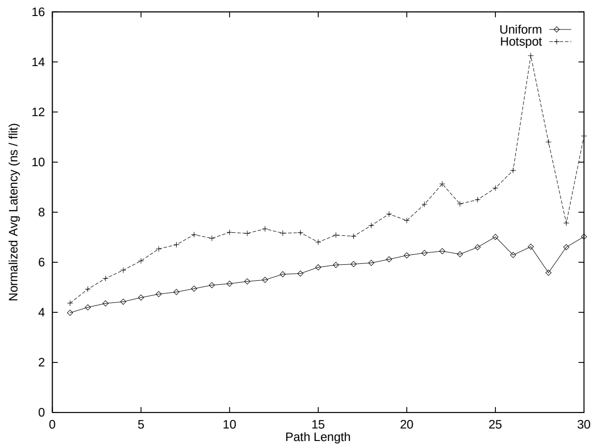
To study the effect of message length on the average and standard deviation of latencies, we designed experiments in which a constant amount of data is pumped into the network, hence the measure "message length $\times$ number of messages injected per unit time", is kept constant. It can be seen, in general, that as message length increases, the average latency per flit reduces, and so does the standard deviation (see Figure 2). This implies that it is preferable to send a longer message, rather than sending out a number of shorter ones.

At low and medium traffic, long messages suffer approximately the same contention. This is reflected in the nearly flat regions in the corresponding curves in Figure 2. On the other hand, at high traffic, a decrease in the average and standard deviation with increase in message length can be observed, because longer messages are generated at a much lower rate than the shorter ones, resulting in lower channel contention. Furthermore, since shorter messages are generated at a higher rate than longer messages, the effect of message startups might also be involved for the shorter messages, leading to a further degradation in performance beyond what has been observed in our experiments.

7.1.3 Effect of Hotspots

As shown in Figure 1, the average and the standard deviation are higher in the presence of hotspots as compared to uniform traffic. This effect can be attributed to the increased blocking of messages as the contention for channels is much higher around the hotspots. The region around each hotspot which has high network traffic saturates first eventually leading to the network getting saturated. Not surprisingly, the network, in the presence of hotspots, saturates earlier than under uniform traffic.

7.1.4 Effect of Path Length



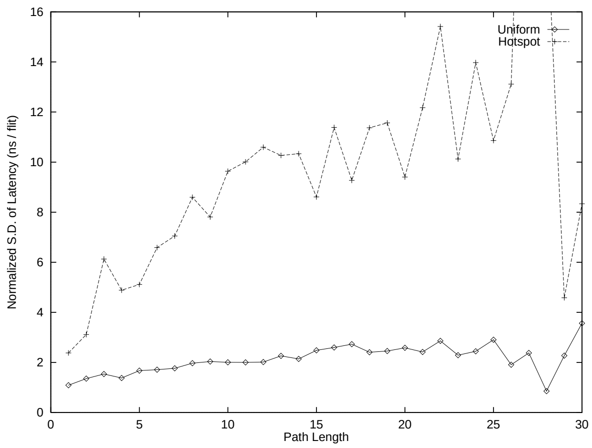


Figure 3: Normalized Latency Measures vs. Path Length for XY routing on a 16x16 mesh with 128-flit messages, for an injection rate of 9.8 flits/cycle

A criticism of using only a few measures such as *overall* average and standard deviation of latencies could be that the measures are crude. Hence, we look at the average and the standard deviation of messages for each path length i ($1 \leq i \leq$ diameter of network). Simulations have revealed that the average and the standard deviation increase with path length. For most path lengths, the standard deviation is in the range of 25% to 50% of the average, as was observed for the overall standard deviation in Section 7.1.1. Hence, similar observations for the need to reduce the standard deviation can be made here too. A representative graph for the behavior of these measures is shown in Figure 3 showing that the problem of standard deviation is severe for messages which travel the same distance.

7.2 Latency Measures – Prioritizing Messages

In this section, we look at the effect of prioritizing messages on the overall latency measures. The results are shown in Figures 4 and 5, for uniform traffic and hotspot traffic, respectively. The effect of additional router design complexity on network cycle time have not been considered in these graphs.

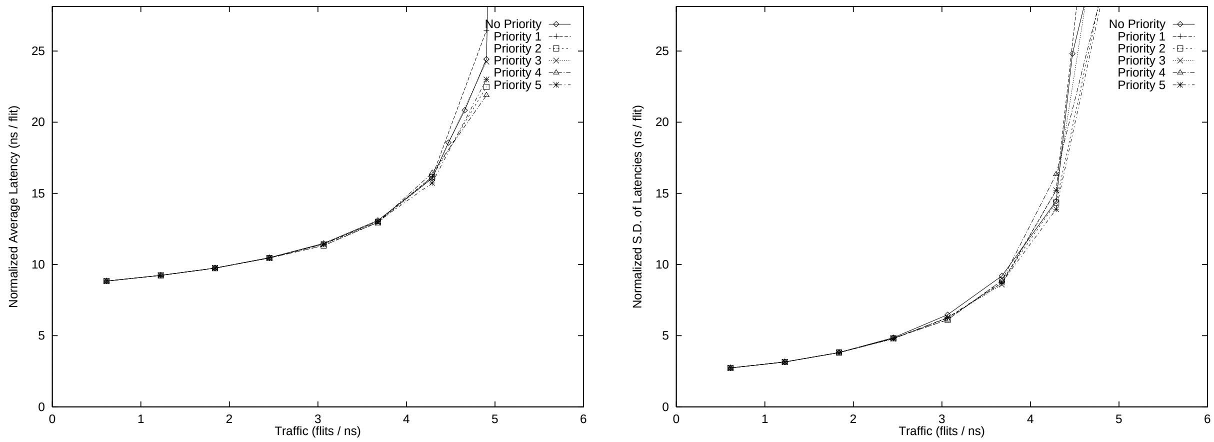


Figure 4: Normalized Latency Measures vs. Traffic for XY routing under Uniform traffic on a 16x16 mesh with 16-flit messages: (Priority 1) Blocking Delay; (Priority 2) Closest First; (Priority 3) Distance Traveled; (Priority 4) Closest-First with ties broken by Blocking Delay; (Priority 5) Normalized Blocking Delay

It can be seen that the *normalized blocking_delay* scheme performs the best. However, the improvement observed with this scheme is less than 8 to 9 % of the average and standard deviations

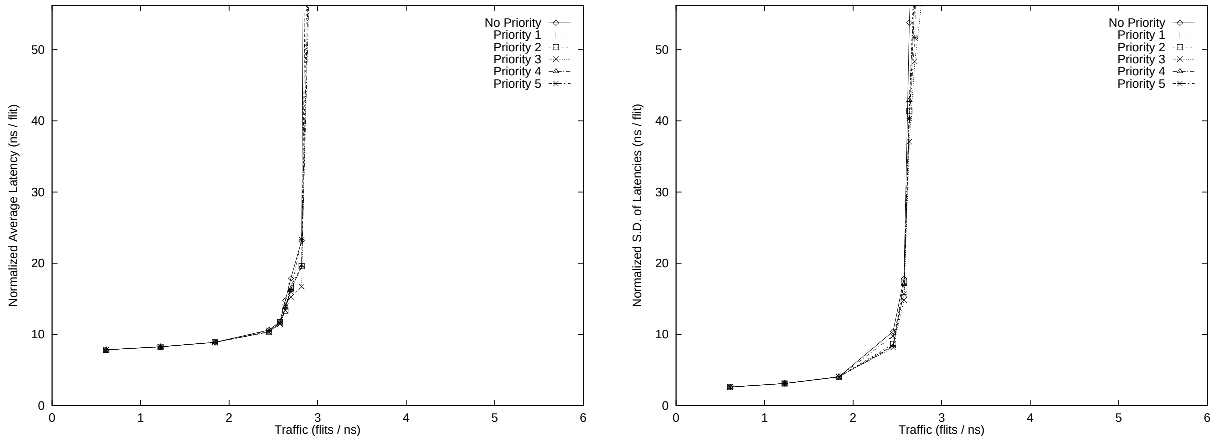


Figure 5: Normalized Latency Measures vs. Traffic for XY routing under Hotspot traffic on a 16x16 mesh with 16-flit messages: (Priority 1) Blocking Delay; (Priority 2) Closest First; (Priority 3) Distance Traveled; (Priority 4) Closest-First with ties broken by Blocking Delay; (Priority 5) Normalized Blocking Delay

without any priority scheme.

With full utilization of network, we did not expect the average to improve with priority schemes. We did expect the standard deviation to improve, however. But, the observed results indicate that the use of priority is not a complete solution by itself. At low network traffic, there is little channel contention, and hence the priorities do not become effective. On the other hand, at high network traffic, the channel reservation by messages in transit, and the consequent blocking of other messages at other routers because of the nature of wormhole routing reduces the effectiveness of priority schemes. The priority comes into effect only for the set of message headers waiting on an output channel at a router when that particular channel becomes available. Owing to this, low priority messages at a router may be blocking high priority messages at other routers. In other words, *the priority schemes exploit local knowledge at a router, but do not take into consideration any global knowledge.* For example, consider a message A that has a very high priority owing to a large delay that it has incurred so far, but somewhere further on its path, another message, say B, has reserved a channel that A requires in order to proceed. Further, assume that B has just been generated and thus has a very low priority. So, it will advance through the rest of the network very slowly. But, it is blocking a high priority message. Other messages that A has blocked are also incurring delays, but they cannot move. So, a low priority message can get delivered whereas several high priority messages are blocked, waiting for the channel to become available.

Also, when the tail of a high priority message, say C, releases a channel, a waiting low priority message, say D, might get that channel if it is the only one blocked there, even though another high priority message, say E, is following the tail of C. Since E incurs header processing time at the node, it does not contend with D for the outgoing channel. This could occur at any node on the path, and could occur repeatedly too, further reducing the beneficial effect of priority schemes on the standard deviation.

Since the use of message priorities does not improve performance even when the hardware complexity is ignored, we have not investigated the performance considering the hardware complexity required by the priority schemes.

7.3 Latency Measures – Multiple Virtual Channels and Adaptivity

In this section, we first look at the effects of using multiple virtual channels and adaptivity, using the traditional "cycle counting" approach, i.e., we ignore the router design complexity associated with virtual channels and adaptivity. Then, we look at the result of using the "cycle time" approach, i.e., we consider the extension of network cycle time by the additional hardware overheads associated with the use of virtual channels and adaptivity. In the following graphs, results have been presented for routing algorithms using one physical channel in each direction; only the number of virtual channels has been varied. The label "XY:i VCS" indicates X-Y routing with i virtual channels per physical channel.

7.3.1 Ignoring router design complexity

The graphs showing the effects of multiple virtual channels with X-Y routing on the latency measures are shown in Figures 6 and 7, for uniform traffic and hotspot traffic, respectively. At low network traffic, messages face little channel contention and hence, the time-multiplexing effectively reduces the bandwidth available to messages, thereby leading to a degradation in performance. But at higher network traffic, it results in an improvement in performance both in terms of the average

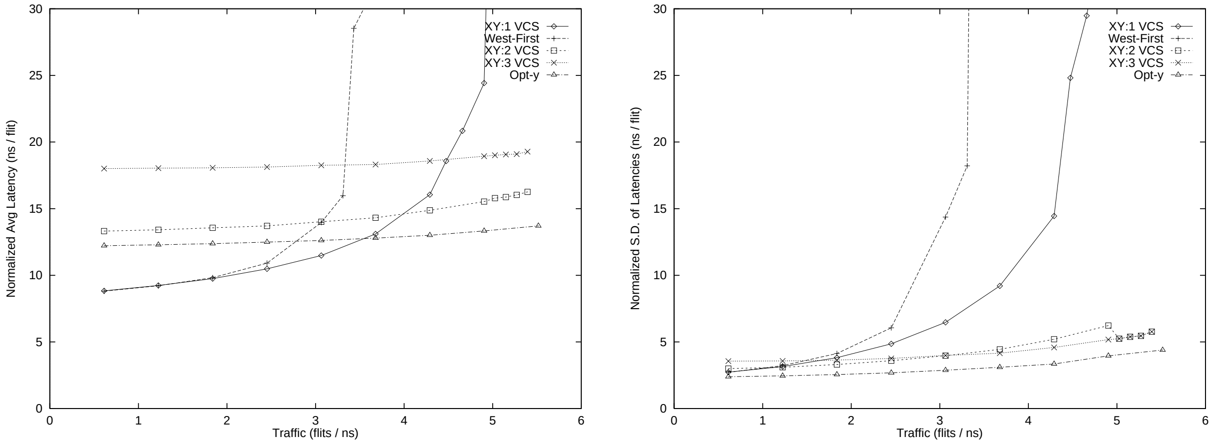


Figure 6: Latency Measures vs. Traffic on a 16x16 mesh under Uniform traffic with 16-flit messages (ignoring hardware overheads)

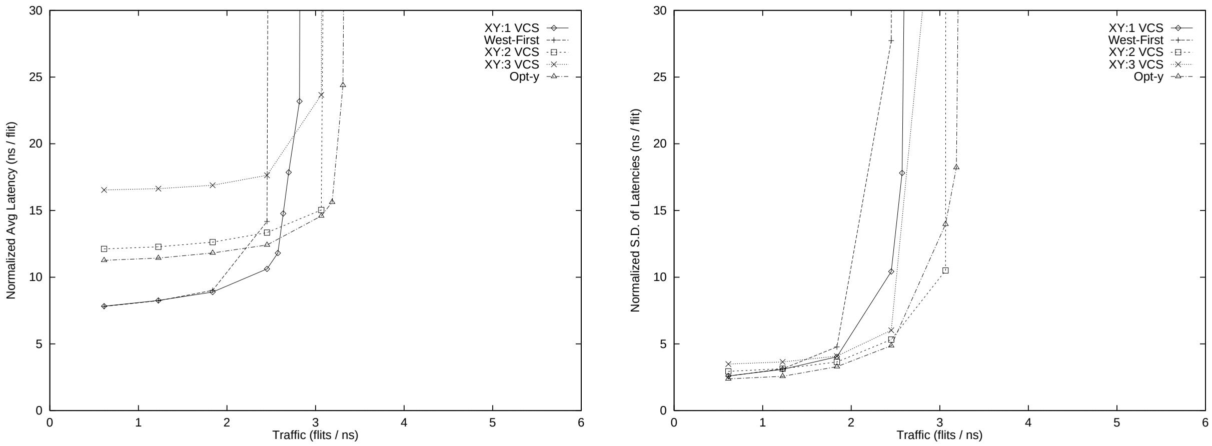


Figure 7: Latency Measures vs. Traffic on a 16x16 mesh under Hotspot traffic with 16-flit messages (ignoring hardware overheads)

and standard deviation, since there is a higher contention for the channels and the use of multiple virtual channels is able to reduce it effectively. Opt-y also leads to an improvement in performance at high network traffic. However, the performance of West-first is similar to that of X-Y at low traffic, but is much worse at high traffic — this is consistent with the observations made by Glass and Ni in [9].

7.3.2 Considering router design complexity

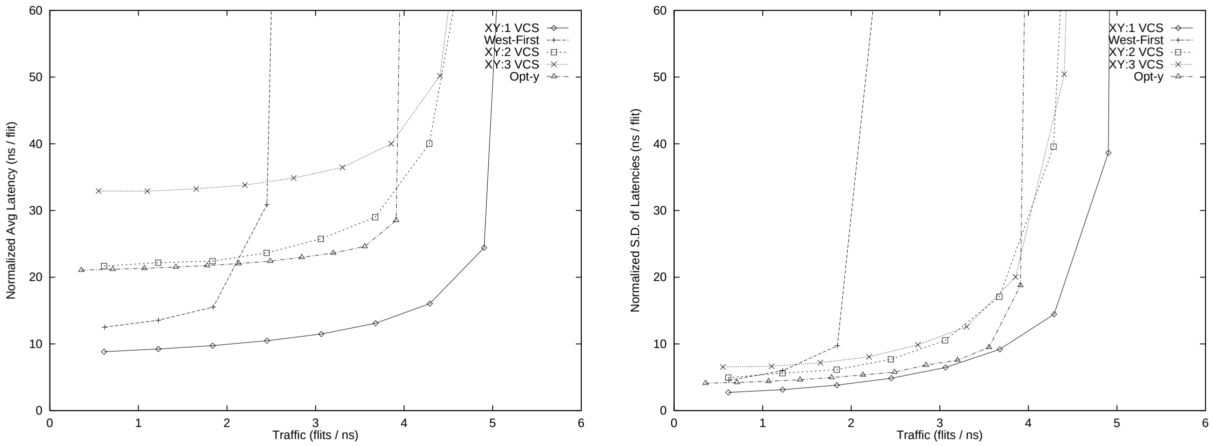


Figure 8: Latency Measures vs. Traffic on a 16x16 mesh under Uniform traffic with 16-flit messages (considering hardware overheads)

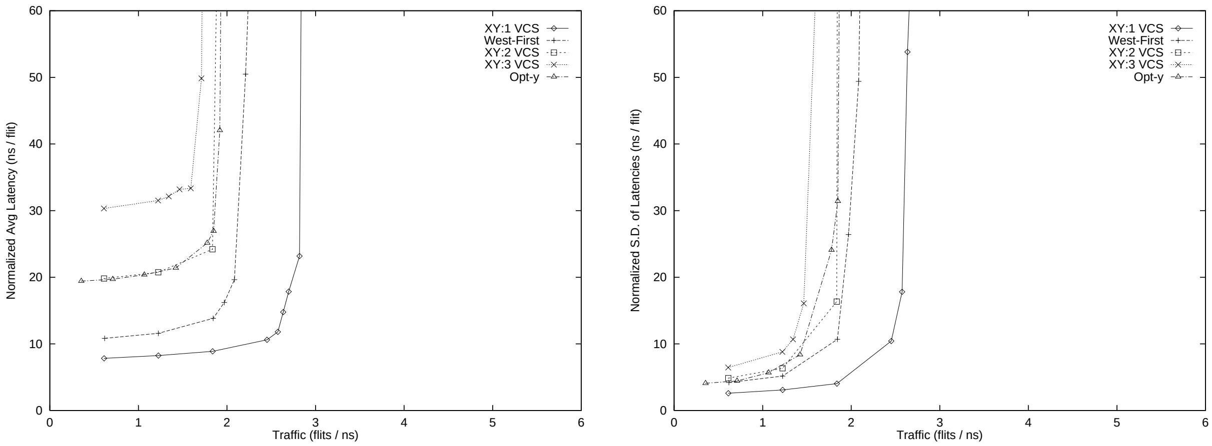


Figure 9: Latency Measures vs. Traffic on a 16x16 mesh under Hotspot traffic with 16-flit messages (considering hardware overheads)

These results are shown in Figure 8 and 9, for uniform traffic and hotspot traffic, respectively. It can be seen that for all the routing algorithms considered, the performance degradation at low network traffic is worse than in Figure 6 and 7. At higher traffic too, the use of virtual channels degrades performance, and the network saturates at a lower load than the base case.

The average latency for X-Y with two virtual channels is almost twice the average for the base case, because of the coupled effect of a network cycle time ratio of 1.64 and time-multiplexing; (similar observation can be made for X-Y with three virtual channels). Note that since the traffic in terms of flits per unit time is the same, *the effect of a higher cycle time for X-Y with multiple virtual channels is to increase the network traffic in terms of flits per cycle. Observe how misleading the previous conclusions were when we had ignored the effects of cycle time degradation.*

7.4 Channel Utilization

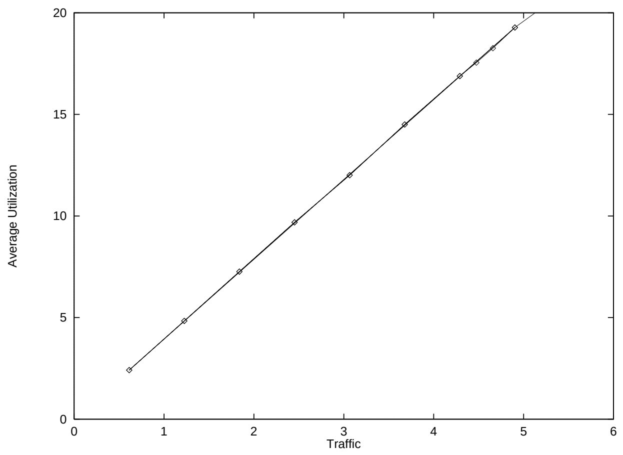


Figure 10: Avg Channel Utilization vs. Traffic for XY routing under Uniform traffic on a 16x16 mesh with 16-flit messages

In all our graphs, we have chosen to represent the traffic in the network using a normalized measure, i.e., the number of flits injected into the network per nanosecond. To put it in perspective with respect to results presented in past literature, we have shown the value of average channel utilization corresponding to each traffic value. The network utilization matches roughly with that observed by other researchers.

8 Conclusions and Future Directions

The standard deviation of message latencies is an important performance measure in wormhole routed MPPs, and must be considered along with the average latency measure. We show that the overall standard deviation and the standard deviations for individual path lengths are high. We further show that it is preferable to use longer messages in order to decrease both the average and the standard deviation. The use of long messages may not always be possible, however, if computation and communication have to be overlapped.

Surprisingly, the use of message priorities, additional virtual channels, or adaptive routing algorithms does not reduce the standard deviation when the associated router design complexity is considered. These schemes, however, give an improvement when the router complexity is ignored, i.e., when the latency ise measured in terms of network cycles rather than the actual time. This shows the drawbacks of the traditional “cycle counting” approach.

We are currently investigating other approaches to reduce the standard deviation of message latencies without adversely affecting the average latency. As we have argued earlier, reducing standard deviation in a multi-user, multiprogrammed MPP environment and in real-time parallel processing is important. We then plan to extend the study to 3-D mesh and torus architectures. One of the constraints in such situations is that decisions which need to be made using global knowledge have to be taken based on local knowledge at the router.

One area that merits investigation is the effect of the multiplexing scheme on message latency. Our focus in this paper has been on time multiplexing. Demand multiplexing the virtual channels seems to be an attractive alternative to time multiplexing. Our preliminary investigation, reported in [11], shows that with demand multiplexing there is a appreciable reduction in the average message latency though the standard deviation remains roughly the same, compared to time multiplexing.

9 Acknowledgments

The authors would like to thank Jeff May and Roshan Rao for help with the implementation of the simulator, and Loren Schwiebert for valuable discussions which have improved the quality of the paper.

References

[1] A. A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Hot Interconnects '93*, August 1993.

[2] W. J. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[3] W. J. Dally and C. L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1(4):187–196, October 1986.

[4] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[5] J. Duato. Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Evaluation of a New Algorithm. In *Symposium on Parallel and Distributed Processing*, pages 840–847, 1991.

[6] J. Duato. Improving the Efficiency of Virtual Channels with Time-Dependent Selection Functions. In *Parallel Architectures and Languages Europe 92*, pages 635–650, 1992.

[7] C. Glass and L. M. Ni. Adaptive Routing in Mesh-Connected Networks. In 12^{th} *International Conference on Distributed Computing Systems*, pages 12–19, 1992.

[8] C. Glass and L. M. Ni. Maximally Fully Adaptive Routing in 2D Meshes. In *International Conference on Parallel Processing*, volume I, pages 101–104, 1992.

[9] C. Glass and L. M. Ni. The Turn Model for Adaptive Routing. In *Proceedings of the* 19^{th} *Annual International Symposium on Computer Architecture*, pages 278–287, 1992.

[10] S. Konstantinidou. Priorities in nonminimal, adaptive routing. In *International Conference on Parallel Processing*, pages I–67–71, 1992.

[11] J. A. May, D. N. Jayasimha, and K. C. Patel. Comparison of Multiplexing Schemes for Wormhole-Routed Networks. In *1st International Workshop on Parallel Processing*, pages 211–215, December 1994.

[12] R. Mraz. Reducing the Variance of Point-to-Point Transfers for Parallel Real-Time Programs. *IEEE Parallel and Distributed Technology*, 2(4):20–31, Winter 1994.

[13] L. Schwiebert and D. N. Jayasimha. Optimal Fully Adaptive Wormhole Routing for Meshes. In *Supercomputing '93*, pages 782–791, 1993.

[14] T. H. Szymanski. On Interconnection Networks for Parallel Processors. *Ph.D Thesis, Dept. of Electrical Engineering, University of Toronto*, 1988.