Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks

Wu-chang Feng and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
E-mail: {wuchang,kgshin}@eecs.umich.edu

Abstract

Maximizing overall performance in multicomputers requires matching application communication characteristics with a suitable routing scheme. However, since the communication demands of emerging applications vary significantly, it is hard for a single routing algorithm to perform well under all workloads. In order to study the complex dependencies between routing policies and communication workloads, we have performed a set of multi-factor experiments to better characterize routing performance. These experiments show that in addition to adaptivity, the selection functions used to order the candidate links greatly affect network performance under various traffic patterns. By supporting flexible routing, the network can tune its routing policies to application communication characteristics in order to improve performance.

Keywords: Multicomputers, interconnection networks, routers, routing

1 Introduction

Message-passing multicomputers are an effective platform for exploiting parallelism in a variety of applications. Since fast message exchange enables efficient, fine-grained cooperation between processing elements, the network used to connect processors must be designed to meet the communication requirements of its applications. Maximizing network performance requires matching application communication characteristics with a suitable network design. Parallel applications impose a wide range of communication patterns on the underlying interconnection network. Scientific computations [1, 2], parallel databases, and real-time applications [3, 4] generate distinct distributions for message lengths, interarrival times, and target destination nodes.

Finding a suitable network design to support such diverse communication requirements is difficult since there are a myriad of design factors which can potentially impact performance. Network topology, network size, the routing algorithm, the switching scheme, and the router

The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the NSF.

architecture all greatly influence the cost and the performance of the design. In this paper, we focus on how routing policies impact network performance as the communication pattern is varied. Specifically, we classify different routing algorithms in terms of their adaptivity and the selection function [5] used to determine the order in which candidate links are considered. Through extensive multi-factor experiments, we investigate how selection functions and the amount of routing adaptivity combine with the traffic pattern to determine how well links are utilized and hence, how well the network performs. These experiments show that no single set of routing policies performs best under all conditions and that tuning network routing to the applied workload can significantly improve network performance. Additional experiments show the utility of supporting multiple routing policies *simultaneously* in a multi-user environment.

The next section of the paper highlights application communication patterns and gives an overview of the various classes of routing algorithms. Section 3 describes the simulation platform used in the experiments. Section 4 evaluates the performance of different selection functions under various application workloads and network topologies. The impact that adaptivity has on these selection functions is then examined in Section 5. Using these results, Section 6 shows the utility of supporting multiple routing schemes by tailoring routing to each application workload to improve performance. Section 7 concludes the paper.

2 Background

2.1 Communication Workloads

Parallel applications generate a wide range of communication workloads depending on the application's granularity and mapping across multiple nodes. Multi-user systems exacerbate these effects since different applications may run *simultaneously*; these applications may execute on different parts of the network or even time-share the same processing elements. Consequently, communication characteristics such as message interarrival times, lengths, and target destinations vary substantially on modern multicomputers, as discussed below.

Message/packet arrival: Earlier studies of multicomputer networks have typically modeled message arrivals as a Poisson process, with exponentially-distributed interarrival times. However, detailed measurements of multicomputer applications have led to more sophisticated message generation models. In particular, these studies show that applications typically generate bursty network traffic [1, 2], due to multi-packet messages and fine-grain handshaking between cooperating nodes.

Message/packet length: Message and packet lengths depend on several factors including packet-size restrictions and the mixture of data and control messages. Although fixed-length packets or exponentially-distributed lengths simplify analytic models, recent work shows that real multicomputer applications typically generate *bimodal* packet-length distributions [1, 2]. Recent studies have examined adapting router designs to accommodate bimodal packet-length distributions [6, 7].

Message destination: Message destination distributions vary a great deal depending on the network topology and the application's mapping onto different processing elements. While many analytical and simulation studies evaluate a uniform random distribution of destination nodes, this pattern does not capture the communication locality or traffic non-uniformities that arise in many applications. Hop-uniform traffic distributions can represent spheres of spatial locality, but these still do not capture the communication structure of specific parallel algorithms or applications. In particular, many scientific programs generate permutation patterns such as matrix-transpose (dimension-reversal), bit-complement, and bit-reversal [8–12]. In this paper, we focus on characterizing network performance over a range of destination distributions.

2.2 Routing

The routing algorithm determines which path a packet takes to reach its destination. Each time a packet enters a node, the routing algorithm generates a list of candidate links for the packet to travel on. *Minimal-path* algorithms generate candidate links along shortest paths while *non-minimal* or *deflection* algorithms can consider additional links in the hope of circumventing network congestion or faulty links. The number of candidate links generated depends on whether or not the algorithm is oblivious or adaptive. If the routing algorithm is oblivious, a single candidate link is chosen deterministically for the incoming packet to route on. Adaptive algorithms, on the other hand, consider multiple outgoing links depending on the prevailing network conditions. While most oblivious algorithms are minimal-path, i.e., route along only minimal path directions, adaptive algorithms can either be minimal or non-minimal. The choice of adaptivity has a significant influence on the cost and performance of the design. By considering multiple outgoing links, adaptive algorithms can increase the likelihood of cut-through at intermediate nodes and balance the traffic load in the network. However, opportunities for adaptive routing vary depending on the network topology and the distance a packet must travel. In addition, adaptive algorithms add to the complexity of both the hardware which implements the scheme and the software which must handle the possibility of out-of-order arrivals [13].

Each algorithm invokes a *selection function* [5] which selects and orders candidate links. Network performance is greatly influenced by the interaction of this function with the communication workload. Selection functions of oblivious algorithms deterministically select a single candidate link independent of the current network conditions. For example, an oblivious random algorithm selects a direction randomly from the candidate links and attempts to route the packet along it. Selection functions of adaptive algorithms determine the order in which the multiple candidate links are considered. For example, an adaptive random algorithm randomly selects from the list of candidate directions until it finds a direction which it can successfully route the packet on.

This paper examines how message destination distributions affect routing algorithm performance. In particular, we focus on how routing adaptivity and selection functions combine with a set of destination distributions to determine network performance. While other studies have looked at comparing various routing algorithms over different patterns, most studies limit the selection functions used [14–16], the range of algorithms evaluated [7, 10, 17], or the destination distributions considered [11, 18].

Distribution	Description
NodeUniform	Uniform random selection of destination node
MatrixTranspose	Source (x, y) selects destination (y, x)
BitComplement	Destination node id is the bit-complement of the source id
BitReversal	Destination node id is the bit-reversal of the source id

Table 1: Destination node distributions

Selection Function	Description
Dimension order	Favors links in lower dimensions of the topology
Random	Generates all rankings with equal probability
Diagonal	Favors directions with more remaining hops

Table 2: Selection functions used in algorithms

3 Experimental Setup

In order to evaluate these algorithms, we used `pp-mess-sim` [19, 20] (point-to-point message simulator), an object-oriented discrete-event simulation tool. `pp-mess-sim` supports a wide range of routing algorithms under a variety of switching schemes by decoupling them from the router models which execute them. The simulator provides a general framework for evaluating router architectures and includes a high-level router model that supports a range of queueing, arbitration, and flow-control policies and can be used to consider a broad range of simulation parameters. For the simulations in this study, the model was configured with word-width crossbar interconnects between reception and transmission links (for cut-through switching schemes), between reception links and host reception ports, and between host transmission ports and transmission links. In addition, the model assumes that the host is an ideal sink with large buffer capacity. This abstract model allows the basic interaction between selection functions and communication workloads to be studied. Experiments were also done on a cycle level simulation model of SPIDER [21] and showed similar trends in performance.

The destination node patterns we examine include node uniform traffic, as well as patterns found commonly in scientific computations such as the matrix-transpose, bit-complement, and bit-reversal permutations. Table 1 summarizes each of these destination distributions. Although these permutations alone do not capture the communication characteristics of all parallel applications, they are sufficient to show the diverse performance trends which exist in parallel systems.

Under these patterns, we evaluate a range of routing algorithms in order to characterize their performance. Specifically, we consider oblivious minpath algorithms using three different selection functions as shown in Table 2. The oblivious dimension-ordered algorithm chooses the lowest dimension link out of all of the minimal-path links. The oblivious random algorithm chooses a single link from all minimal-path links randomly. Finally, the oblivious diagonal algorithm chooses the link which is in the direction that the packet has the most hops left to travel [22,23].

Similar to the oblivious algorithms, adaptive minpath algorithms using the same selection functions are considered. The adaptive dimension-ordered algorithm takes the set of minimal-path directions, orders them according to their dimension, and attempts to route the packet along each of the ordered directions until it is successful. The random and diagonal algorithms do the same except the directions are ordered randomly and according to the number of hops left, respectively. By choosing the direction which has the most hops left to travel on, the diagonal algorithm attempts to maximize the available routing options for each packet as it travels from source to destination.

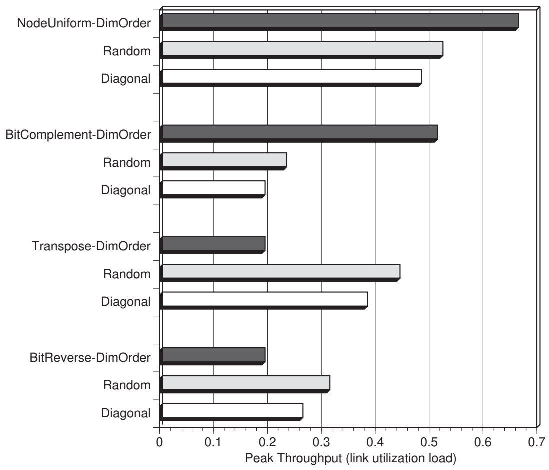
The non-minimal algorithms we evaluate attempt to route in the minimal-path directions first before attempting non-minimal ones. The number of non-minimal hops (deflections) allowed is limited by a hop threshhold in order to prevent livelock. As with the minpath algorithms, deflection algorithms also have associated selection functions. In this paper, we evaluate a class of deflection algorithms which use a dimension-ordered selection function for ordering both the minimal and non-minimal directions. In these experiments, the hop threshhold is varied in order to show the performance of algorithms with varying degrees of adaptivity. Note that these deflection algorithms typically perform close to the adaptive dimension-ordered minpath algorithm, since dimension-ordered minpath routing is effectively the deflection algorithm with a hop threshold of 0. Any benefit that the deflections provide can thus be measured against the performance of this algorithm. For certain patterns, we also consider deflection algorithms which use random and diagonal selection functions for ordering the minimal-path directions.

The experiments were performed on two different, low-dimensional network topologies: the square mesh and the torus. The torus differs from the square mesh in that it has wrapped links which connect nodes on the periphery to each other and make the topology homogeneous. By comparing the performance of the same algorithms over both the torus and square mesh, we examine how topologies impact the performance of adaptivity and selection functions.

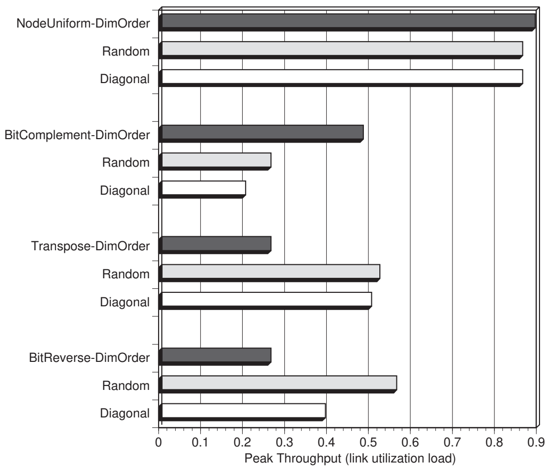
For the experiments in Section 4 and 5, a packet length of 16 words and a network size of 256 nodes were used. Simulations using 64-word packets and 64-node meshes were also performed and showed no significant differences. The results of these experiments are included in Appendix A. While the experiments in these sections also fix the switching scheme as virtual cut-through, experiments using wormhole switching are presented in Section 6.

4 Selection Functions

In order to show how selection functions impact performance, we evaluated oblivious algorithms using each of the selection functions in Table 2. Figure 1 shows the peak throughputs (given in link utilization load) of each algorithm under the different traffic patterns. While the peak throughputs are a good indication of how algorithms behave at lower loads, algorithms which saturate at higher peak throughputs don't necessarily have lower latencies. For such cases, average latency graphs over all link loads are given.



(a) 16 x 16 square mesh



(b) 16 x 16 torus

Figure 1: Peak throughput for oblivious algorithms under different traffic patterns

4.1 Node Uniform Traffic

For node uniform traffic in a square mesh, selection functions have a significant impact on performance. The dimension-ordered algorithm outperforms both the random and diagonal selection functions. This occurs since the diagonal and random algorithms *generate* pockets of congestion by clogging the center of the mesh. The dimension-ordered selection function performs the best since it tends to avoid the middle of the mesh and is able to utilize the links at the periphery of the network more effectively. The diagonal function performs the worst since it leaves these links underutilized by preferring routes which go through the center of the mesh. Finally, since the random selection function neither prevents nor prefers routing toward the center of the mesh, its performance is in between the two. In contrast to the square mesh, the oblivious algorithms perform comparably to each other on a torus. This is because both the topology and the communication pattern, in this case, are homogeneous. Thus, all links remain evenly loaded as long as the selection function used isn't biased.

4.2 Bit-Complement Traffic

Figure 1 also shows the peak throughputs of the oblivious algorithms under bit-complement traffic. This pattern fundamentally congests the center of the network in both the torus and square mesh topologies, leaving many of the peripheral links underutilized. The bit-complement permutation requires source node (c, d) to communicate with node $(15 - c, 15 - d)$; as a result, all packets must eventually cross both the middle row and the middle column of the mesh, irrespective of the routing algorithm. Again, the dimension-ordered algorithm tends to avoid the center of the network, where the middle row and column meet, by exhausting the x-direction before routing a packet in the y-direction. On the other hand, the random and diagonal algorithms perform poorly since they allow packets to route to the center of the mesh, causing the links along the periphery to be even more underutilized. This effect is seen in both

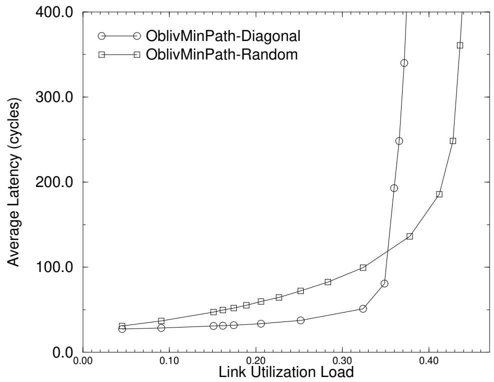


Figure 2: Performance of oblivious random and diagonal algorithms under transpose traffic

the square mesh and the torus experiments.

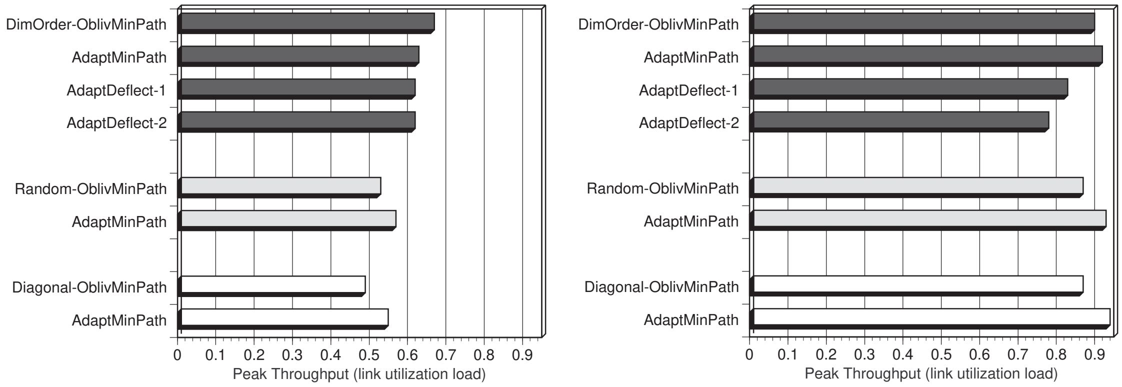
4.3 Matrix Transpose and Bit-Reversal Traffic

The trends in performance change under transpose and bit-reversal traffic. For both patterns, all nodes of a particular row communicate with nodes of a particular column. Using the dimension-ordered algorithm, packets route on common links, leaving a large number of links underutilized throughout the network. Both the random and the diagonal algorithm outperform the dimension-ordered algorithm by taking advantage of these underutilized links. Because the diagonal algorithm statically chooses a common path for all packets between a source-destination pair, its peak throughput is lower than the random algorithm's, as shown in Figure 1. By randomizing its decisions, the random algorithm more evenly distributes the traffic across the links at high loads.

While the random algorithm outperforms the diagonal algorithm at high loads, the diagonal algorithm achieves a lower average latency at low loads as shown in Figure 2 for transpose traffic on a 16×16 square mesh. The diagonal algorithm performs well at low loads since it forces the traffic off of the heavily-loaded links and onto the underutilized links; the random algorithm only partially does this, leaving links along particular rows and columns fairly congested. At higher loads, however, the diagonal algorithm prematurely saturates the diagonal links it routes on since it statically chooses a single path from source to destination.

5 Adaptivity

By considering multiple links, adaptive algorithms can potentially improve performance over oblivious algorithms by routing around regions of congestion. For each pattern, we evaluate



(a) 16 x 16 square mesh (b) 16 x 16 torus

Figure 3: Peak throughput under node-uniform traffic

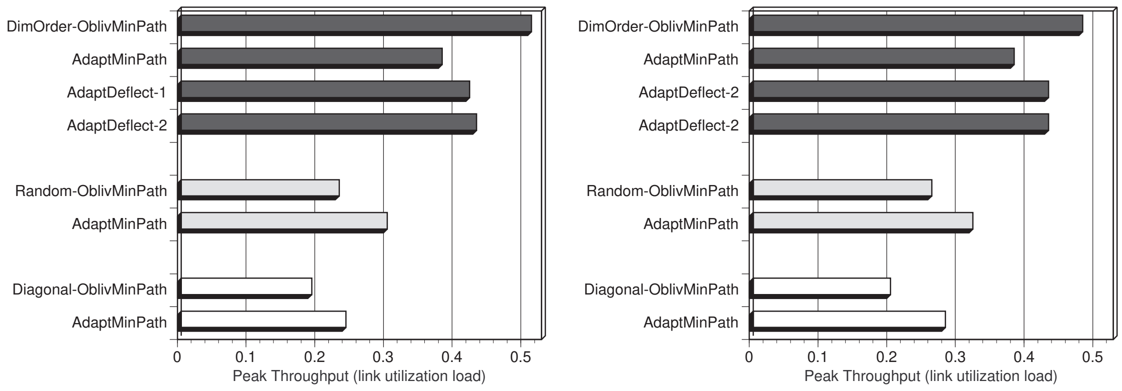
a number of adaptive algorithms using different selection functions to see how they impact routing performance. Each plot groups the algorithms according to their selection function with each group ordered according to the amount of adaptivity. The deflection algorithms are also labeled with their hop threshhold. Additional deflection experiments using larger hop threshholds are shown in Appendix A. The benefits of extra deflections diminish significantly after a hop threshhold of 2.

5.1 Node Uniform Traffic

Figure 3 shows the peak realizable throughputs of the adaptive and oblivious algorithms under node-uniform traffic. For this pattern, adaptivity has little impact on network performance. For the random and diagonal selection functions, adaptivity slightly improves performance since the algorithms can use adaptivity to avoid congestion created by their selection functions. For the dimension-ordered selection function, the performance of adaptive algorithms depends on the topology. On a torus, the dimension-ordered adaptive minpath algorithm only slightly outperforms the oblivious algorithm since opportunities for adaptivity are limited for a homogeneous traffic pattern on a homogeneous topology. In this case, the deflection algorithms actually *reduce* performance since deflections delay packets and consume additional link bandwidth without avoiding much congestion. As Figure 3(b) shows, increasing the hop threshold for the deflection algorithms exacerbates this effect by allowing more (wasteful) misroutes. In a square mesh, the oblivious dimension-ordered algorithm actually *outperforms* the adaptive minpath algorithm. This occurs because any adaptive routes send packets closer to the congested center of the network. Because the deflection algorithms can use their adaptivity to route around this congestion, they do not harm performance as they did with the torus topology.

5.2 Bit-Complement Traffic

Figure 4 shows the peak throughputs of the different algorithms under bit-complement



(a) 16 x 16 square mesh (b) 16 x 16 torus

Figure 4: Peak throughput under bit-complement traffic

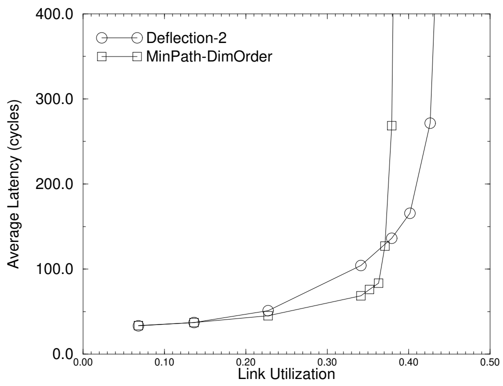
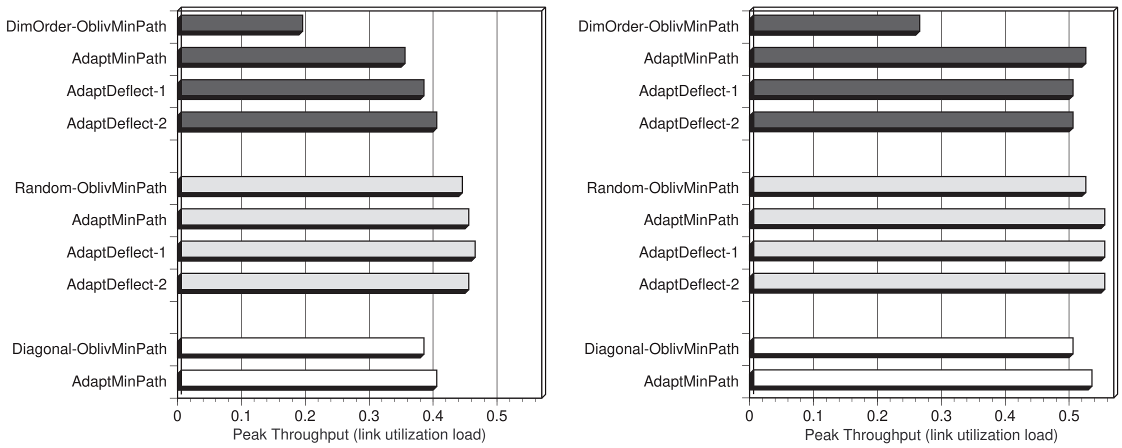


Figure 5: Minimal and nonminimal adaptive routing under bit-complement traffic

traffic. As the figure shows, none of the adaptive algorithms performs as well as the oblivious dimension-ordered algorithm. However, adaptivity does help improve performance of algorithms using the random or diagonal selection functions. Since these functions tend to direct traffic into the center of the network, adaptivity helps by choosing alternate links in order to avoid creating large regions of congestion. For the dimension-ordered selection function, adaptivity harms performance since the adaptive algorithms mistakenly try to avoid the heavily-congested middle column (or row) by routing packets to more lightly-loaded rows (or columns); this ultimately pushes traffic *closer* to the congested center of the network. A local decision at one node causes a packet to travel a lightly-loaded link into a more congested region. This effect becomes worse on a square mesh and on larger networks, where the regions of congestion are magnified.

As shown in Figure 4, deflections can help improve performance over the dimension-ordered adaptive minpath algorithm by circumventing the regions of congestion. Thus, the deflections correct mistakes made earlier in the route from selecting adaptive minimal path links. While



(a) 16 x 16 square mesh (b) 16 x 16 torus

Figure 6: Peak throughput under matrix transpose traffic

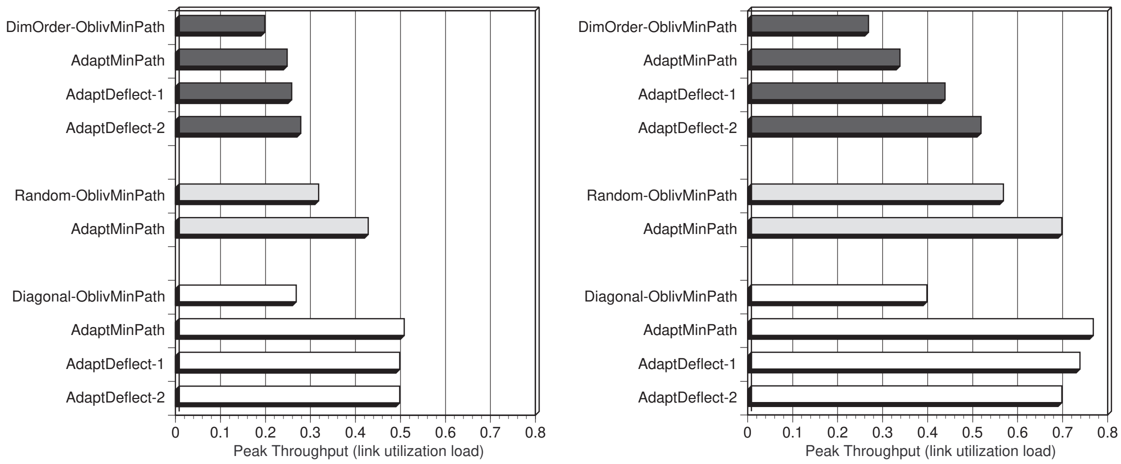
the deflection algorithms outperform the minpath algorithms at high loads, the minimal routing algorithms perform better at low loads, as shown in Figure 5. This graph compares the average packet latency of the dimension-ordered minpath algorithm and the 2-hop deflection algorithm in a 16×16 square mesh network. At lower loads, nonminimal routing increases end-to-end packet latency, since network congestion is not severe enough for deflections to have a positive impact.

5.3 Matrix-Transpose Traffic

Figure 6 shows the peak throughputs of the adaptive and oblivious algorithms under matrix transpose traffic. Adaptivity significantly improves performance for the oblivious dimension-ordered algorithm by routing packets onto underutilized links. In this case, the underutilized links are interspersed throughout the network, enabling the adaptive algorithms to find and utilize them. This is in contrast to the bit-complement traffic pattern, where the underutilized links are at the periphery of the network. For the diagonal and random algorithms, however, adaptivity makes little difference since the selection function itself is enough to distribute the load onto the underutilized links.

As Figrue 6 shows, regardless of the amount of adaptivity, algorithms using a random selection function peform the best. This occurs because any of the deterministic selection functions bias certain paths over others and generates pockets of congestion. Since the random selection function does not prefer a particular routing direction, it evenly distributes the traffic across the network, and thus saturates at a higher load.

The behavior of the dimension-ordered deflection algorithms varies with the network topology. While deflections improve performance (relative to the adaptive minpath algorithm) for the square mesh topology, they hinder performance in the torus. This occurs because the square mesh generates more non-uniformity in the network, thus giving the deflections more opportunities on which to capitalize. Since the random selection function performs best for the



(a) 16 x 16 square mesh

(b) 16 x 16 torus

Figure 7: Peak throughput under bit-reversal traffic

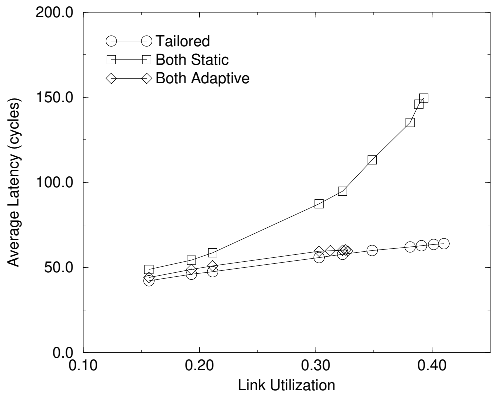
minpath algorithms, experiments using deflection algorithms with random selection functions were also performed. The additional adaptivity, however, made little difference in performance as shown in Figure 6.

5.4 Bit-Reversal Traffic

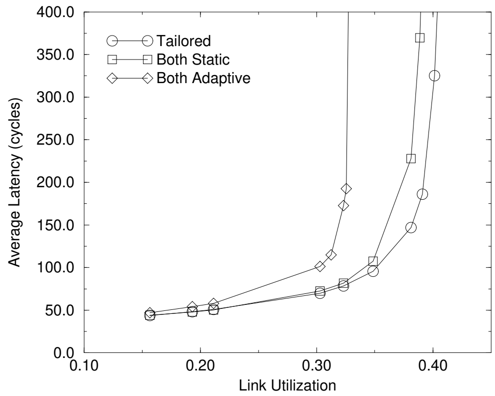
Figure 7 shows the peak throughputs of the routing algorithms under a bit-reversal traffic pattern. As with the matrix-transpose pattern, oblivious dimension-ordered routing performs poorly since packets can not circumvent regions of congestion to utilize many of the lightly-loaded links. However, additional adaptivity, in this case, does not improve performance significantly as shown by the low peak throughputs of the adaptive minpath and adaptive deflection dimension-ordered algorithms. This indicates that just as the diagonal selection function is pathologically bad for bit-complement traffic, the dimension-ordered function is pathologically bad for bit-reversal traffic.

Unlike the matrix-transpose pattern, simply changing selection functions is not enough to obtain peak performance. As shown in Figure 7, the adaptive minpath algorithms using random and diagonal selection functions perform considerably better than any of the oblivious algorithms. It is interesting to note that the adaptive diagonal algorithm saturates at a higher peak throughput than the adaptive random algorithm, while the reverse is true for their oblivious counterparts. For the adaptive algorithms, the diagonal selection function attempts to maximize the number of nodes in which the packet has two possible directions to route on. Since this gives the packet more of a chance to find lightly-loaded links later in its route, the algorithm outperforms the adaptive random algorithm.

The performance of the deflection algorithms depends on the selection function. Deflections improve performance for the dimension-ordered deflection algorithms since they allow packets to route around the areas of congestion caused by the poor performance of the selection function.



(a) Bit-reversal traffic



(b) Bit-complement traffic

Figure 8: Average latency under traffic mixing

This effect increases with the hop-threshold. Since the diagonal selection function performed the best for the minpath algorithms, additional experiments using a diagonal deflection algorithm were also performed. As Figure 7 shows, the extra adaptivity makes little difference and actually harms performance in some cases.

6 Tailoring Experiments

The multi-factor experiments show that the relative performance of routing algorithms varies according to application communication characteristics. Because of the diverse communication requirements of emerging applications, several router architectures now support multiple routing schemes in order to tailor network routing to applications [4, 24–26]. In a multi-user environment, where multiple tasks communicate concurrently, supporting multiple routing schemes *simultaneously*, can improve performance significantly.

The following experiements consider a mix of bit-reversal and bit-complement traffic in an 8×8 square mesh. The experiments evaluate a wormhole network with four virtual channels on each link; the two tasks route messages on separate pairs of virtual channels to partially insulate the applications from each other. According to Figures 4(a) and 7(a), bit-complement has peak performance under oblivious dimension-ordered routing, while the bit-reversal pattern performs best under the diagonal minpath algorithm. Using these results, we evaluate three configurations: one where both tasks use the oblivious algorithm, one where both tasks use the adaptive algorithm, and finally one where the bit-complement task uses the oblivious algorithm and the bit-reversal task uses the adaptive one.

Figure 8 shows average packet latency for both traffic patterns, under increasing bit-complement load; the bit-reversal pattern remains fixed at a link load of 0.12. As shown in Figure 8(a), the bit-reversal traffic has poor performance when both tasks are forced to use the oblivious routing algorithm. Bit-reversal performance improves significantly when both tasks employ diagonal minpath routing, but this configuration degrades the bit-complement performance, as shown in Figure 8(b). The bit-complement traffic has low average latency under oblivious dimension-ordered routing, independent of the algorithm assigned to the bit-reversal traffic. In this case, the network performs best when it tailors the routing policies to the application traffic patterns.

7 Conclusion

Our experiments have shown that, under a range of application workloads, the performance of routing algorithms varies significantly. In particular, routing performance is fairly sensitive to the selection function, adaptivity, and the traffic pattern.

Applying these results to improve routing performance can easily be done for systems such as HARTS [27], the nCube-3 [25], and Hnet [24] which support multiple routing policies. For example, in HARTS, the Programmable Routing Controller (PRC) [19,26] can be reprogrammed with different routing policies as the application communication workloads change. By having multiple algorithms programmed into its memory, the PRC is also able to tailor its routing decisions to individaul tasks in multi-user environments.

For systems which only support a single routing policy, these results can be used to influence how the operating system maps the tasks onto multicomputer nodes, in effect, tailoring the communication workload to the routing algorithm. Future work will address such issues as well as examine performance under more realistic application workload models such as bursty sources and bimodal packet lengths.

References

[1] J.-M. Hsu and P. Banerjee, "Performance measurement and trace driven simulation of parallel CAD and numeric applications on a hypercube multicomputer," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, pp. 451–464, July 1992.

[2] R. Cypher, A. Ho, S. Konstantinidou, and P. Messina, "Architectural requirements of parallel scientific applications with explicit communication," in *Proc. Int'l Symposium on Computer Architecture*, pp. 2–13, May 1993.

[3] J.-P. Li and M. W. Mutka, "Priority based real-time communication for large scale wormhole networks," in *Proc. International Parallel Processing Symposium*, pp. 433–438, April 1994.

[4] J. Rexford and K. G. Shin, "Support for multiple classes of traffic in multicomputer routers," in *Proc. Parallel Computer Routing and Communication Workshop*, pp. 116–130, May 1994.

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

[6] S. Konstantinidou, "Segment router: A novel router design for parallel computers," in *Symposium on Parallel Algorithms and Architectures*, June 1994.

[7] J. H. Kim and A. A. Chien, "Evaluation of wormhole routed networks under hybrid traffic loads," in *Proc. Hawaii Int'l Conf. on System Sciences*, pp. 276–285, January 1993.

[8] S. Chittor and R. Enbody, "Performance evaluation of mesh-connected wormhole-routed networks for interprocessor communication in multicomputers," in *Supercomputing*, pp. 647–656, November 1990.

[9] J. H. Kim and A. A. Chien, "An evaluation of planar-adaptive routing (PAR)," in *Proc. International Symposium on Parallel and Distributed Processing*, 1992.

[10] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," *Journal of the ACM*, vol. 42, pp. 91–123, January 1995.

[11] W. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, pp. 466–475, April 1993.

[12] S. Ramany and D. Eager, "The interaction between virtual channel flow control and adaptive routing in wormhole networks," in *Proc. International Conference on Supercomputing*, pp. 136–145, July 1994.

[13] V. Karamcheti and A. A. Chien, "Do faster routers imply faster communication?," in *Proc. Parallel Computer Routing and Communication Workshop*, pp. 1–15, June 1994.

[14] R. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," in *Proc. Int'l Symposium on Computer Architecture*, pp. 351–360, 1993.

[15] T. D. Nguyen and L. Snyder, "Performance analysis of a minimal adaptive router," in *Proc. Parallel Computer Routing and Communication Workshop*, pp. 31–44, May 1994.

[16] J. Duato and P. Lopez, "Performance evaluation of adaptive routing algorithms for k-ary n-cubes," in *Proc. Parallel Computer Routing and Communication Workshop*, pp. 45–59, May 1994.

[17] T. Nesson and L. Johnsson, "Romm routing: A class of efficient minimal routing algorithms," in *Proc. Parallel Computer Routing and Communication Workshop*, pp. 185–199, May 1994.

[18] P. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks," *IEEE Computer*, pp. 12–23, May 1993.

[19] J. Dolter, *A Programmable Routing Controller Supporting Multi-mode Routing and Switching in Distributed Real-Time Systems.* PhD thesis, University of Michigan, September 1993.

[20] J. Rexford, J. Dolter, W. Feng, and K. G. Shin, "PP-MESS-SIM: A simulator for evaluating multicomputer interconnection networks," in *Proc. Simulation Symposium*, pp. 84–93, April 1995.

[21] J. Dolter, S. Daniel, A. Mehra, J. Rexford, W. Feng, and K. Shin, "SPIDER: Flexible and efficient communication support for point-to-point distributed systems," in *Proc. Int'l Conf. on Distributed Computing Systems*, pp. 574–580, June 1994.

[22] H. G. Badr and S. Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE Trans. Computers*, vol. C-38, pp. 1362–1370, October 1989.

[23] A. L. Davis, "Mayfly: A general-purpose, scalable, parallel processing architecture," *Lisp and Symbolic Computation*, vol. 5, pp. 7–47, May 1992.

[24] D. Smitley, F. Hady, and D. Burns, "Hnet: A high-performance network evaluation testbed," Tech. Rep. SRC-TR-91-049, Supercomputing Research Center, Institute for Defense Analyses, Dec. 1991.

[25] nCube Corporation, *nCUBE-3: The Scalable Server Platform*, March 1995.

[26] S. Daniel, J. Rexford, J. Dolter, and K. Shin, "A programmable routing controller for flexible communications in point- to-point networks." to appear in *Proc. Int'l Conference on Computer Design*, October 1995.

[27] K. G. Shin, "HARTS: A distributed real-time architecture," *IEEE Computer*, vol. 24, pp. 25–35, May 1991.

Appendix A Simulation Results

Tables 3-8 give the peak throughputs points for different routing algorithms in terms of link utilization load applied. For each entry, a percent is given which indicates how far below the best routing algorithm, a given algorithm performs. This inidicates the relative performance between each of the algorithms as parameters are varied.

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.51 (0%)	0.19 (-62%)	0.19 (-59%)	0.66 (0%)
Diagonal	0.19 (-63%)	0.26 (-48%)	0.38*(-17%)	0.48 (-27%)
Random	0.23 (-55%)	0.31 (-38%)	0.44*(- 4%)	0.52 (-21%)
AdaptMinPath-DimOrder	0.38 (-25%)	0.24 (-52%)	0.35 (-24%)	0.62 (- 6%)
Diagonal	0.24 (-53%)	0.50 (0%)	0.40 (-13%)	0.54 (-18%)
Random	0.30 (-41%)	0.42 (-16%)	0.45 (- 2%)	0.56 (-15%)
AdaptDeflect-DimOrder-1	0.42 (-18%)	0.25 (-50%)	0.38 (-17%)	0.61 (- 8%)
DimOrder-2	0.43 (-16%)	0.27 (-46%)	0.40 (-13%)	0.61 (- 8%)
DimOrder-3	0.43 (-16%)	0.30 (-40%)	0.42 (- 9%)	0.60 (- 9%)
DimOrder-4	0.43 (-16%)	0.31 (-38%)	0.42 (- 9%)	0.59 (-11%)
Diagonal-1		0.49 (- 2%)		
Diagonal-2		0.49 (- 2%)		
Random-1			0.45 (- 2%)	
Random-2			0.46 (0%)	

Table 3: 16×16 Square Mesh, 16 word packets

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.48 (0%)	0.26 (-66%)	0.26 (-53%)	0.89 (- 4%)
Diagonal	0.20 (-58%)	0.39 (-49%)	0.50 (- 9%)	0.86 (- 8%)
Random	0.26 (-46%)	0.56 (-26%)	0.52 (- 5%)	0.86 (- 8%)
AdaptMinPath-DimOrder	0.38 (-21%)	0.33 (-57%)	0.52 (- 5%)	0.91 (- 2%)
Diagonal	0.28 (-42%)	0.76 (0%)	0.53 (- 4%)	0.93 (0%)
Random	0.32 (-33%)	0.69 (- 9%)	0.55 (0%)	0.92 (- 1%)
AdaptDeflect-DimOrder-1	0.43 (-10%)	0.43 (-43%)	0.50 (- 9%)	0.82 (-12%)
DimOrder-2	0.43 (-10%)	0.51 (-33%)	0.50 (- 9%)	0.77 (-17%)
DimOrder-3	0.43 (-10%)	0.52 (-32%)	0.49 (-11%)	0.73 (-22%)
Diagonal-1		0.73 (- 4%)		
Diagonal-2		0.69 (- 9%)		
Random-1			0.55 (0%)	
Random-2			0.55 (0%)	

Table 4: 16×16 Torus, 16 word packets

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.54 (0%)	0.22 (-58%)	0.21 (-60%)	0.72 (0%)
AdaptMinPath-DimOrder	0.44 (-19%)	0.32 (-40%)	0.47 (-11%)	0.68 (- 6%)
Diagonal	0.32 (-41%)	0.53 (0%)	0.45 (-15%)	0.59 (-18%)
Random	0.36 (-33%)	0.49 (- 8%)	0.53 (0%)	0.63 (-13%)
AdaptDeflect-DimOrder-1	0.48 (-11%)	0.33 (-38%)	0.46 (-13%)	0.69 (- 4%)
DimOrder-2	0.48 (-11%)	0.37 (-30%)	0.48 (- 9%)	0.68 (- 6%)
DimOrder-3	0.49 (- 9%)	0.43 (-19%)	0.48 (- 9%)	0.66 (- 8%)
DimOrder-4	0.49 (- 9%)	0.42 (-21%)	0.47 (-11%)	0.65 (-10%)

Table 5: 8×8 Square Mesh, 16 word packets

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.47 (0%)	0.27 (-58%)	0.28 (-53%)	0.91 (- 1%)
AdaptMinPath-DimOrder	0.42 (-11%)	0.41 (-37%)	0.54 (-10%)	0.92 (0%)
Diagonal	0.38 (-19%)	0.60 (- 8%)	0.57 (- 5%)	0.92 (0%)
Random	0.38 (-19%)	0.65 (0%)	0.60 (0%)	0.92 (0%)
AdaptDeflect-DimOrder-1	0.45 (- 4%)	0.48 (-26%)	0.52 (-13%)	0.85 (- 7%)
DimOrder-2	0.45 (- 4%)	0.50 (-23%)	0.54 (-10%)	0.82 (-11%)
DimOrder-3	0.45 (- 4%)	0.48 (-26%)	0.51 (-15%)	0.80 (-13%)

Table 6: 8×8 Torus, 16 word packets

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.51 (0%)	0.20 (-61%)	0.20 (-53%)	0.63 (0%)
AdaptMinPath-DimOrder	0.38 (-25%)	0.26 (-49%)	0.37 (-14%)	0.58 (- 8%)
Diagonal	0.24 (-53%)	0.51 (0%)	0.43 (0%)	0.51 (-19%)
AdaptDeflect-DimOrder-1	0.42 (-18%)	0.27 (-47%)	0.40 (- 7%)	0.58 (- 8%)
DimOrder-2	0.43 (-16%)	0.29 (-43%)	0.43 (0%)	0.58 (- 8%)

Table 7: 16×16 Square Mesh, 64 word packets

Algorithm	BitComplement	BitReverse	Transpose	Uniform
OblivMinPath-DimOrder	0.48 (0%)	0.27 (-61%)	0.28 (-51%)	0.90 (- 3%)
AdaptMinPath-DimOrder	0.39 (-19%)	0.36 (-49%)	0.56 (- 2%)	0.93 (0%)
Diagonal	0.28 (-42%)	0.82 (0%)	0.57 (0%)	0.93 (0%)
AdaptDeflect-DimOrder-1	0.43 (-10%)	0.47 (-47%)	0.54 (- 5%)	0.83 (-11%)
DimOrder-2	0.43 (-10%)	0.54 (-43%)	0.54 (- 5%)	0.78 (-16%)

Table 8: 16×16 Torus, 64 word packets