A Comparison of Input and Output Driven Routers [†]

Melanie L. Fulgham
Lawrence Snyder
{mel, snyder}@cs.washington.edu
University of Washington

November 27, 1996

Abstract

Communication in parallel computers requires a low latency router. Once a suitable routing algorithm is selected, an implementation must be designed. Issues such as whether the router should be input or output driven need to be considered. In this paper, we use simulations to compare input driven and output driven routing algorithms. Three algorithms, the Dally-Seitz oblivious router, the *-channels router, and the minimal triplex algorithm are evaluated. Each router is implemented as both an input and an output driven router. Experiments are run for each of the router implementations with seven different traffic patterns on both a 256-node two dimensional mesh and torus networks. The results show that in almost all cases, the output driven router matches or outperforms the input driven router. Furthermore, we find that randomization of output buffer selection in the input driven algorithm increases its performance and substantially reduces the performance discrepancy between the input and output driven algorithms. Although the findings apply to the routers considered, we believe the results generalize to other routers.

1 Introduction

Communication in parallel computers continues to be a very difficult problem. In this paper we consider communication in multicomputer networks, networks with point to point connections between processors. In this model, processors communicate by sending messages through the network. Messages are forwarded at each node to their destination by a hardware router that implements the routing algorithm, the rules of which specify the path a message takes to reach its destination. Initially the communication bottleneck was the large

[†]This work supported in part by NSF Grant MIP-9213469 and by an ARPA Graduate Research Fellowship. A shorter version of this paper appears in Lecture Notes in Computer Science, volume 1123, pages 195-204, 1996.

software overhead for sending and receiving messages. Nevertheless, research has reduced this overhead [vEDCGS92, Dal90, Kea94] and exposed the network interface. Designing low latency network interfaces has become a hot topic for research [BLA+94, MBES94]. Eventually network interfaces will no longer overshadow the routing time in the network. Then, both the routing algorithm and its implementation will have an impact on communication performance.

In this paper, we experimentally compare two methods of implementing a particular router, as an input driven or as an output driven algorithm. Although the two choices are conceptually similar, they result in noticeable performance differences when compared with each other. Three algorithms, the Dally-Seitz oblivious router, the *-channels router, and the minimal triplex router, are simulated on a 256-node two dimensional (2D) mesh and torus. Each algorithm is configured as both an input and an output driven router. The output driven versions of the algorithms are almost always superior when the networks are congested. This is the critical area of network performance, since communication is often bursty and does not always present easy, light loads to the network.

The paper is organized as follows. Section 2 defines input and output driven routers. The simulation methodology and the three routing algorithms are described in Section 3. Results follow in Section 4, related work in Section 5, and finally conclusions in Section 6.

2 Input versus Output Driven

Routers that make decisions based on local information can usually be classified as either input driven or output driven. The input driven algorithms make routing decisions from an occupied input buffer while the output driven algorithms select routes from an empty output buffer. Typically the input driven algorithms service the input buffers in round robin order, and the output driven algorithms service the output buffers in round robin order, After routing, the pointer to the current buffer is advanced to the next dimension where an interesting buffer resides. For input driven algorithms an interesting input buffer is one that contains a ready message that needs an available output buffer. For output driven algorithms an interesting output buffer is one that is empty and is wanted by some ready input message.

An input driven router operates as follows. First it computes for the current input message which output buffers the message needs. Then it selects one of the available output buffers and routes the message to that output buffer. Many of the routing algorithms in the literature fall into this category. See Figure 1 for an example of an input driven router.

An output driven algorithm considers the current empty output buffer. The router tries to find a message in an input buffer that needs to be routed to the current output buffer. If messages are found, it selects one to be routed to the current output buffer. The Chaos router is a good example of an output driven router [KS94]. See Figure 2 for an example of an output driven router.

Many algorithms can be implemented as either input or output driven algorithms. For these algorithms, it would be advantageous to implement the router with the best performing

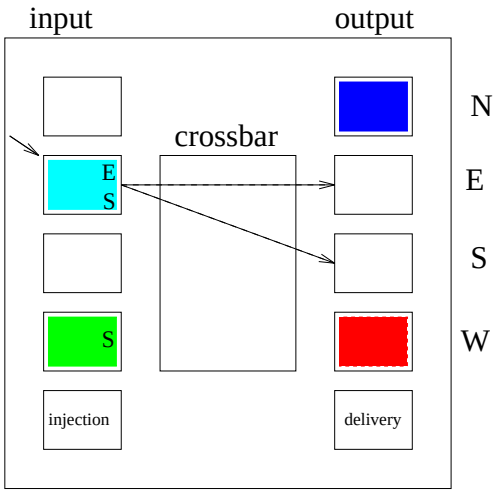

Figure 1: An example of an input driven router. The short arrow marks the current input buffer. The long arrows are the choices of empty output buffers for the message in the current input buffer. Messages can travel north (N), east (E), south (S), or west (W). Buffers that are filled in are not available, while the unfilled buffers are available.

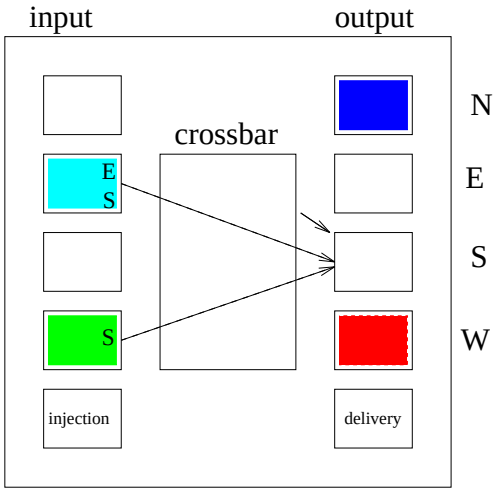

Figure 2: An example of an output driven router. The short arrow marks the current output buffer. The long arrows are from messages that are competing for the current empty output buffer. Messages can travel north (N), east (E), south (S), or west (W). Buffers that are filled in are not available, while the unfilled buffers are available.

routing mechanism. A later section shows that for the algorithms examined, the output driven algorithm generally performs as well as, or better than the input driven algorithm. Unfortunately some algorithms cannot easily be transformed into output driven algorithms, for example algorithms that take probabilities over the possible output choices.

3 Methodology

Performance of input versus output driven routing algorithms is compared by experimentation. Three algorithms are simulated as both input and then output driven routers: the Dally-Seitz oblivious router [DS87], the *-channels router [BGPS92, Dua93], and the minimal triplex router[1] [FS96] using a flit-level simulator of 256-node 2D torus and mesh networks.

A node (x, y) in a 2D torus has 4 neighbors: $(x + 1 \bmod k, y)$, $(x - 1 \bmod k, y)$, $(x, y + 1 \bmod k)$, and $(x, y - 1 \bmod k)$, where k is the size of a dimension. Edges between nodes of the form $(k - 1, y)$ and $(0, y)$ or $(x, k - 1)$ and $(x, 0)$ are called *wrap* edges. A mesh is a torus without wrap edges.

The first algorithm, the Dally-Seitz oblivious router, provides no adaptivity and routes messages by correcting the dimensions from lowest to highest (e.g. in the 2D case, x then y). Unless specified, the oblivious router uses two virtual lanes [Dal92] resulting in four virtual channels per channel for the torus and two virtual channels for the mesh. When more than one lane is available, the one with the most space is chosen. The *-channels algorithm is a minimal fully adaptive algorithm which uses three (two) virtual channels per channel for the torus (mesh). For the torus (mesh), two (one) of these virtual channels are (is) used for oblivious dimension order routing and the other is used for minimal routing. The minimal triplex router is a minimal fully adaptive version of the triplex router. It also uses three (two) virtual channels per channel for the torus (mesh). The torus (mesh) algorithm provides minimal adaptive routing on all three (two) virtual channels. All the algorithms have routing restrictions that prevent deadlock.

The following describes the high level design and operation of the routers. Each router has an injection buffer, a delivery buffer, and an input and an output buffer for each virtual channel in each dimension in each direction. This yields 26 (18) buffers per node for the *-channels and triplex routers on the torus (mesh), since each uses an injection buffer, a delivery buffer, and three (two) virtual channels per channel per direction. The oblivious router is configured with four (two) virtual channels per channel per direction for the torus (mesh) or 34 (18) buffers per node.

Transmission of a flit over a channel from an output buffer to a neighboring node's input buffer costs one cycle. The actual cycle time depends on the technology used for implementation. Decoding and routing calculations are pipelined to allow the router cycle time to match that of the channel. The more complex algorithms require a larger pipeline depth. This study uses node latencies of three cycles for the oblivious router, and four cycles

[1] An early implementation of the triplex router is used. Since then, performance has improved, though the differences between the input and output driven routers should remain unchanged.

for the adaptive routers [Bol93]. To keep complexity manageable, the router connects at most a single message from an input buffer to an output buffer per cycle.

Traffic is dynamic. Messages are introduced at each node at every cycle with a probability specified by the applied load. For clarity, the applied load is normalized by the maximum sustainable load when an average of half the messages cross the network bisection[2], as with uniform random traffic. Thus, for a kxk-node network with an average message length of l flits, the maximum possible load is 1 message every $kl/4$ cycles for the torus and 1 message every $kl/2$ cycles for the mesh. Messages are 20 flits long and buffers can hold an entire message exactly. In this case for a 256-node 2D torus (mesh), the maximum normalized applied load (1.0) corresponds to the applied load of one message every 80 (160) cycles. Virtual cut-through flow control is used [KK79]. This avoids store-and-forward latency penalties. With virtual cut-through, a buffer may contain parts of two distinct messages.

The traffic patterns considered are found in the literature, and are generally thought to be difficult, useful, or both. The following describes the traffic patterns simulated. Let the binary representation of the source node be $a_{n-1}a_{n-2}\ldots a_0$, and let $\overline{0} = 1$ and $\overline{1} = 0$.

- Random - all destinations including the source are equally likely.
- Bit Reversal permutation - destination is $a_0 a_1 \ldots a_{n-1}$.
- Complement permutation - destination is to $\overline{a_{n-1}a_{n-2}\ldots a_0}$.
- Perfect Shuffle permutation - destination is $a_{n-2}a_{n-2}\ldots a_0 a_{n-1}$.
- Transpose permutation - destination is $a_{n/2-1}a_{n/2-2}\ldots a_0 a_{n-1}a_{n-2}\ldots a_{n/2}$.
- Hot spot - ten randomly selected nodes are four times more likely to be chosen as destinations than the other nodes.

For the hot spot traffic, two different configurations were simulated. Assuming the nodes are labeled in row major order from 0 to 255, the hot spot nodes for case 1 are 158, 186, 216, 236, 121, 86, 6, 152, 201, and 123. For case 2 they are 51, 92, 254, 140, 51, 70, 201, 155, 124, and 245.

The traffic patterns illustrate different features. As mentioned earlier the random traffic is simply a standard benchmark used in network routing studies. The hot spot traffic models cases where references to program data, such as synchronization locks, bias packet destinations towards a few nodes. The complement is a particularly difficult permutation. Given an imaginary x and y axis though the center of a mesh or torus network, the complement destination is the composition of the x and y axis reflection of the source. Perfect shuffle communication occurs in ascend/descend algorithms [PV81] while the transpose and bit reversal are important because they occur in practical computations.

The simulator is written in C and uses a batch means method [Muñ91] for computing 95% confidence intervals for the expected values of network throughput and latency. The sources

[2]The network bisection is the minimum number of channels cut to divide the network in half.

of randomness are provided by a prime-modulus, multiplicative congruential generator [LL74] which is considered highly reliable for simulation studies [LO89].

4 Results

The results are presented in two parts. The first set of experiments compare output driven routers to input driven routers that choose deterministically the first available output buffer from the set of needed output buffers. Any deterministic order suffices, in this case the routers search in dimension order. In the second set of experiments the input driven router selects an available output buffer at random from the set of needed output buffers. For this set-up, the input driven router is very similar to the output driven router which chooses messages from the input buffers at random. In both cases the output driven routers perform as well as or better than the input driven routers for all but a few cases. As expected, the results are more dramatic for the first case.

4.1 Fixed Order Output Buffer Selection

The first set of experiments consider input driven routers with a fixed order output buffer selection policy. Table 1 specifies the first normalized applied load, in increments of .05, at which messages arrive faster then they can be injected and delivered by the 256-node torus network. For all three algorithms and all traffic patterns except one, the complement, the point of saturation for the output driven router is at least as great as the saturation point of the corresponding input driven router. The advantage ranges from zero to fifty-four percent of the input driven saturation point.

Table 1: Minimum normalized applied load within .05 at which saturation is detected.

16x16 Torus Saturation, input driven is fixed order						
Traffic	oblivious		*-channels		min triplex	
	input	output	input	output	input	output
Random	0.80	0.80	0.85	0.95	0.80	0.85
Bit reversal	0.50	0.50	0.70	0.80	0.65	0.75
Complement	0.50	0.50	0.45	0.40	0.40	0.40
Perfect shuffle	0.50	0.50	0.50	0.50	0.40	0.45
Transpose	0.55	0.55	0.55	0.55	0.55	0.55
Hot Spot 1	0.65	0.65	0.70	0.90	0.65	0.85
Hot Spot 2	0.55	0.55	0.55	0.85	0.60	0.80

We conjecture that the output driven algorithms are better at balancing the load among the channels because the empty output buffers are filled by dimension in round-robin order which naturally tries to keep all the physical channels busy. More specifically, when a message has been routed to an empty output buffer in a dimension, the router will consider the next

dimension that has an empty output buffer. The input driven algorithm, however, routes a message to the first available output buffer it finds. This buffer may share the same physical channel as a recently routed message. This cannot happen in an output driven router unless all the other output buffers in the other dimensions are full, or all the messages in the input buffers only need dimensions that are being used by previously routed messages.

Table 2 presents the saturation loads for the 256-node mesh and shows that output driven routers are superior to input driven routers for the mesh also. The advantage ranges from zero to thirteen percent. Again, the only exception is the complement traffic for the *-channels and minimal triplex adaptive routers.

Table 2: Minimum normalized applied load within .05 at which saturation is detected.

16x16 Mesh Saturation, input driven is fixed order								
Traffic	oblivious (no vc)		oblivious		*-channels		min triplex	
	input	output	input	output	input	output	input	output
Random	0.90	0.90	0.95	0.95	0.95	0.95	0.90	0.90
Bit reversal	0.50	0.50	0.55	0.55	0.80	0.80	0.70	0.75
Complement	0.45	0.50	0.50	0.50	0.45	0.35	0.45	0.35
Perfect shuffle	0.75	0.80	0.90	0.90	0.90	0.95	0.80	0.90
Transpose	0.50	0.50	0.55	0.55	0.85	0.85	0.85	0.85
Hot Spot 1	0.75	0.75	0.80	0.80	0.80	0.85	0.75	0.85
Hot Spot 2	0.70	0.70	0.75	0.75	0.75	0.85	0.75	0.85

We expect the benefit of output driven routers to be less pronounced or non-existent in oblivious routers than in adaptive ones, since the two oblivious routers make the same routing decisions. The oblivious routers may, however, move a message from an input to an output buffer at a different time or select a different lane to route a message to, though the latter is unlikely to effect performance since lanes share the same underlying physical channel. The data supports this hypothesis since there is almost no difference between the saturation points of the input and the output driven oblivious routers on the torus and mesh.

Routing the complement permutation using the *-channels algorithm is the one case where the output driven saturation point did not equal or exceed the input driven saturation load for either topology. The complement is unusual since dimension order routing helps prevent conflicts in this traffic pattern, i.e. when two messages compete for the same output buffer.

To see this, consider the following idealized scenario. Each node simultaneously injects a single message destined for the bit complement of its source id. When routing is by a synchronous, oblivious dimension order algorithm, there are no conflicts for channels, as explained in the following. With dimension order routing, every message m with source (i, j) is routed in row i in exactly the same way as message m' with source (k, j) is routed in row k. All messages move within the rows in lock step. Conflicts can only occur when a messages turns from its source row to its destination column. Messages from column j are the only messages that need to travel in destination column $\bar{j}$. Messages m and m' arrive

at their destination column $\bar{j}$ at the same time, and hence do not impede each other from progressing to their respective destinations. Arbitrary injections are used in our routing experiments, however, and they perturb the orderly flow of messages. Adaptive routing further destroys this property by letting messages take any minimal path.

For the *-channels algorithm the input driven router has a slight advantage in exploiting this phenomenon since it prefers the lowest dimension output buffer needed, while the output driven algorithm selects a random message that needs the current output buffer. Nevertheless, the second set of experiments show this advantage disappears when the input driven algorithm picks from the needed output buffers at random.

For the mesh, the oblivious algorithm was also simulated with no virtual channels (vc), i.e. with no extra lanes. In this case, a message waiting in an input buffer needs exactly one output buffer. Therefore, the input and output driven algorithms do not make different routing decisions or buffer choices. Rather, the difference in performance is due to the ordering of messages and the router's ability to keep the channels busy. As hypothesized, the difference in saturation points between input and output driven routers is very modest, non-existent in five traffic patterns and within a normalized load of .05 for the two remaining ones. For the two lane oblivious routers, there is no difference in the saturation points between the input and output driven routers.

Figures 3–8 show the expected throughput and latency versus the normalized applied load for each of the traffic patterns on the 256-node torus and mesh. Throughput represents messages delivered per cycle and is normalized to reflect bandwidth limitations of the network. Latency measures time in the network and does not include source queueing, since after saturation, the source queue length is not well defined.

At low loads the input and output driven routers are indistinguishable. Nevertheless, the output driven router achieves a higher or equivalent peak throughput than the corresponding input driven router for all but one of the traffic patterns and routers simulated. The peak throughput of the output driven router on the torus (mesh) is up to 36 (13) percent better than the input driven router with fixed order selection. See Tables 5–10 for details. As with the saturation data, the complement traffic is the only exception. For the minimal triplex router on the mesh and torus, the peak throughput of the output driven router under the complement traffic does not quite reach that of the input driven router. The same is true for *-channels on the mesh.

After saturation, the output driven router almost always maintains a higher throughput than the input driven router, though there are three exceptions. The output driven router degrades slightly more than the input driven router for a few of the applied loads for the perfect shuffle and second hot spot case with the oblivious router on the torus and for the second hot spot traffic with *-channels on the mesh.

The latency curves have three phases. Initially the input and output schemes have equivalent latencies. Any router and routing decision will do when the router is lightly loaded. When the applied load is in the neighborhood of saturation and the network is congested, the output driven routers almost always exhibit a lower or equivalent latency (and latency variance) than the input driven router. This is apparent by observing that

the output driven routers experience a steep increase in latency at the same or higher load than the input driven routers. We believe this is because the output driven router is doing a better job at keeping the physical channels utilized. There are two exceptions. The adaptive output driven routers show an increase in latency at a slightly lower load than the input driven router for the complement on the torus and mesh and for random traffic on the mesh.

After saturation the network cannot keep up with the message arrivals; and again, any type of routing will do. Furthermore, the latency differences are less predictable, though they tend to converge for many of the traffic patterns. In two instances, bit reversal and transpose on the torus, the oblivious input driven router shows a decrease in its after saturation latency. In these cases, many fewer messages with distant destinations are injected into the network, thereby lowering the expected latency of a message. This appears to be caused by the unfair access each node has to a congested network. A similar phenomena occurs with bit reversal traffic at a load of .8 using input driven triplex routing.

4.2 Random Output Buffer Selection

To validate our conjecture that the advantage of output driven routers is not from the non-deterministic search for available output buffers, the following changes were made to make the input driven routers as similar as possible to the output driven routers. Each input driven router was modified so that it selects a needed output buffer at random, instead of choosing the lowest dimension output buffer available. The output driven routers were unchanged and choose messages from the input buffers at random. Experiments showed the modified input driven algorithms to have better channel utilization resulting in improved performance[3].

Tables 3 and 4 compare the saturation points of the input and the output driven routers for the torus and mesh. The change from fixed order to random output buffer selection does not change the buffer selection of the oblivious routers, and hence the results for input driven routing with random selection are the same as those with fixed order buffer selection.

For the adaptive routers the difference between the saturation points of the input and output driven algorithms on the mesh and torus has nearly been eliminated for almost all the traffic patterns. The output driven advantage ranges from zero to 14 percent of the input driven saturation point for the torus and to 13 percent for the mesh. The input driven routers with random output buffer selection are superior to those with fixed order output selection, since removing the bias of fixed order selection improves the utilization of the physical channels of a node. The only exception is the complement. The complement saturates at a lower load than previously for both adaptive routers on the mesh and torus. This is because the fixed order input selection prefers the dimension order route if available; and as mentioned earlier, the complement prefers dimension order routing.

For the mesh hot spot traffic, the *-channels input driven router with random selection has a higher saturation point than the output driven router, even though the output driven

[3]It is possible that round-robin selection could approximate this improvement without the use of randomization, since deterministically spreading the outgoing messages among the various channels would still help channel utilization.

Table 3: Minimum normalized applied load within .05 at which saturation is detected.

16x16 Torus Saturation, input driven is random				
Traffic	*-channels		min triplex	
	input	output	input	output
Random	0.95	0.95	0.85	0.85
Bit reversal	0.80	0.80	0.70	0.75
Complement	0.40	0.40	0.35	0.40
Perfect shuffle	0.50	0.50	0.45	0.45
Transpose	0.55	0.55	0.55	0.55
Hot Spot 1	0.85	0.90	0.80	0.85
Hot Spot 2	0.75	0.85	0.75	0.80

Table 4: Minimum normalized applied load within .05 at which saturation is detected.

16x16 Mesh Saturation, input driven is random				
Traffic	*-channels		min triplex	
	input	output	input	output
Random	0.95	0.95	0.90	0.90
Bit reversal	0.80	0.80	0.75	0.75
Complement	0.35	0.35	0.35	0.35
Perfect shuffle	0.95	0.95	0.90	0.90
Transpose	0.80	0.85	0.75	0.85
Hot Spot 1	0.90	0.85	0.85	0.85
Hot Spot 2	0.90	0.85	0.85	0.85

router nearly achieves the same throughput. The input driven router is able to prevent congestion around the hot spots from spreading as quickly into the whole network compared to the output driven router. The reasons for this are unclear, but most likely related to the lack of wrap edges in the mesh since the torus does not exhibit this behavior.

Figures 9–14 compare the expected throughput and latency of the input and output driven algorithms on the torus and mesh. The oblivious comparisons are identical to the previous ones.

The adaptive input driven routers improve their maximum achieved throughput, and in some cases now match the throughput of the corresponding output driven routers, as with bit reversal and random traffic with *-channels routing and random traffic using triplex routing on the torus. For the mesh, random selection closed the gap between input and output driven performance with triplex on the bit reversal, random, perfect shuffle, and both hot spot cases, as well as for the perfect shuffle traffic pattern with the *-channels algorithm. In addition, random selection sometimes results in less throughput degradation for the adaptive algorithms on the mesh, as with the perfect shuffle traffic pattern.

The output driven router achieves up to an 11 (6) percent greater peak throughput than the input driven router on the torus (mesh). The two exceptions, as with the saturation points, are the hot spot cases where the *-channels input driven algorithm achieves a slightly higher peak throughput than the output driven algorithm. See Tables 5–10 for details.

There are also latency improvements for the adaptive routers with many of the traffic patterns. In these cases, the steep increase in latency occurs at a higher load and after saturation latency is also smaller. In a few cases however, throughput degradation causes an increase in latency. See Figures 15–20 in the Appendix for direct comparisons between the two input driven schemes for both the mesh and torus.

5 Related Work

Most of the work in routing algorithms has been in developing deadlock-free algorithms. Numerous frameworks have been presented for developing deadlock-free algorithms with varying complexity, resource requirements, and switching techniques [DS87, LH91, Dua93, Dua95, GN94, CK92, SJ95, LC94]. Each of these factors influences the overall performance of the router. Nevertheless, only a few studies have been devoted to improving performance or comparing various implementations of a routing algorithm. Dally increased throughput of wormhole algorithms by adding virtual channels to separate the buffering resources from the transmission resources of the router [Dal92]. Konstantinidou reduced overall message latency in bimodal length traffic by introducing segment routing [Kon94]. Segment routing provides a separate buffer for large messages, allowing small messages to pass larger ones. Cherkasova and Rokicki replaced FIFO injection, the traditional method of introducing messages into the network, with alpha scheduling [CR94]. With variable length messages, alpha scheduling approximates the optimal average message latency of shortest first scheduling without introducing starvation. Finally, a study by Glass and Ni compared the performance of various policies for selecting input and output buffers for two different routing algorithms

on the mesh [GN92].

6 Conclusions

We have experimentally compared the performance of input and output driven algorithms on the mesh and torus. Although the two are conceptually similar, for almost all the cases examined, the performance of the output driven algorithms is equivalent, or superior to that of the input driven algorithms. The difference is diminished when randomization is added to the output buffer selection of the input driven algorithm. Although the findings presented only apply to the routers considered, we believe that the results can be generalized to routers where the designer is indifferent to which approach to use. Future work may compare input and output driven routers using longer messages or a non-minimal router.

References

[BGPS92] P.E. Berman, L. Gravano, G. Pifarré, and J.L.C. Sanz. Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks. In *Proc. of the Sym. of Parallel Algorithms and Architectures*, 1992.

[BLA+94] M.A. Blumrich, K. Li, R. Alpert, C. Dubnicki, and E.W. Felten. Virtual memory mapped network interface for the SHRIMP multicomputer. In *Proc. of the Intl. Sym. on Computer Arch.*, pages 142–153, 1994.

[Bol93] Kevin Bolding. *Chaotic Routing: Design and Implementation of an Adaptive Multicomputer Network Router*. PhD thesis, University of Washington, Seattle, July 1993.

[CK92] A.A. Chien and J.H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. In *Proc. of the Intl. Sym. on Computer Architecture*, pages 268–277, 1992.

[CR94] L. Cherkasova and T. Rokicki. Alpha message scheduling for packet-switched interconnects. Technical Report TR HPL-94-72, Hewlett-Packard Labs, 1994.

[Dal90] W.J. Dally. The J-machine system. In P.Winston and S. Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers*. MIT Press, 1990.

[Dal92] W. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[DS87] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[Dua93] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–475, April 1993.

[Dua95] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10), October 1995.

[FS96] M. Fulgham and L. Snyder. Triplex router: a versatile torus routing algorithm. Technical Report UW-CSE-96-01-11, Univ. of Washington, Seattle, 1996.

[GN92] C.J. Glass and L.M. Ni. Adaptive routing in mesh-connected networks. In *Proc. of the Intl. Conf. on Distributed Computing Systems*, pages 12–19, 1992.

[GN94] Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. *JACM*, 41(5):874–902, 1994.

[Kea94] J. Kuskin and et al. The Stanford FLASH multiprocessor. In *Proc. of ISCA*, pages 302–313, 1994.

[KK79] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[Kon94] S. Konstantinidou. The segment router: a novel router design for parallel computers. In *Proc. of the Sym. of Parallel Algorithms and Architectures*, pages 364–373, 1994.

[KS94] S. Konstantinidou and L. Snyder. The chaos router. *IEEE Transactions on Computers*, 43(12):1386–97, December 1994.

[LC94] Z. Liu and A.A. Chien. Hierarchical adaptive routing: A framework for fully adaptive and deadlock-free wormhole routing. In *Sym. on Par. and Distr. Processing*, pages 688–695, 1994.

[LH91] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers*, C-40(1):2–12, January 1991.

[LL74] G.P. Learmonth and P.A.W. Lewis. Statistical tests of some widely used and recently proposed uniform ran dom number generators. In *Proc. of the 7th Conf. on Comp. Sci, and Stats. Interface*, 1974.

[LO89] P.A.W. Lewis and E.J. Orav. *Uniform Pseudo-Random Variable Generation.* Wadsworth Brooks/Cole, 1989.

[MBES94] N. McKenzie, K. Bolding, C. Ebeling, and L. Snyder. CRANIUM: An interface for message passing on adaptive packet routing networks. In *Lecture Notes in Computer Science*, volume 853, pages 266–80, 1994.

[Muñ91] David Muñoz. Multivariate standardized time series in the analysis of simulation out put. Technical Report TR-68, Operations Research, Stanford University, Palo Alto, CA, April 1991.

[PV81] F. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, 1981.

[SJ95] L. Schwiebert and D.N. Jayasimha. A universal proof technique for deadlock-free routing in interconnection networks. In *Proc. of the Sym. on Par. Alg. and Arch.*, pages 175–184, 1995.

[vEDCGS92] T. von Eicken D.E. Culler, S.C. Goldstein, and K.E. Schauser. Active messages: a mechanism for integrated communication and computation. In *Proc. of the Intl. Sym. on Computer Arch.*, pages 256–266, May 1992.

A Appendix

Tables 5–10 contain the peak normalized throughput, rounded to the nearest whole number, achieved by each of the routers for the various traffic patterns.

Table 5: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Torus, oblivious						
Traffic	output			input fixed or random		
	load	xput	error	load	xput	error
Random	0.80	78	0.3	0.80	77	0.4
Bit reversal	1.00	46	0.2	0.60	44	0.3
Complement	0.45	45	0.2	0.45	44	0.3
Perfect shuffle	0.45	45	0.4	0.45	45	0.3
Transpose	0.55	50	0.1	0.60	50	0.1
Hot Spot 1	0.65	63	1.1	0.65	63	1.1
Hot Spot 2	0.50	50	0.2	0.55	51	2.6

Table 6: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Torus, *-channels									
Traffic	output			input fixed			input random		
	load	xput	error	load	xput	error	load	xput	error
Random	0.90	89	0.2	0.80	70	1.3	0.90	90	0.2
Bit reversal	0.75	70	0.2	0.65	61	0.3	0.75	70	0.2
Complement	0.40	39	0.3	0.40	40	0.3	0.40	39	0.4
Perfect shuffle	0.45	45	0.3	0.45	45	0.6	0.45	45	0.3
Transpose	0.55	50	0.1	0.55	50	0.1	0.55	50	0.2
Hot Spot 1	0.85	85	0.4	0.80	70	2.1	0.80	80	0.2
Hot Spot 2	0.80	80	1.2	1.00	59	2.2	0.75	72	1.9

Table 7: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Torus, minimal triplex									
Traffic	output			input fixed			input random		
	load	xput	error	load	xput	error	load	xput	error
Random	0.80	80	0.2	0.75	74	0.2	0.80	80	0.2
Bit reversal	0.70	66	0.2	0.60	56	0.3	0.65	61	0.3
Complement	0.40	36	0.6	0.40	40	0.5	0.35	34	0.5
Perfect shuffle	0.45	44	0.6	0.40	39	1.2	0.40	40	0.3
Transpose	0.55	50	0.1	0.55	50	0.2	0.55	50	0.1
Hot Spot 1	0.80	80	0.3	0.65	61	3.7	0.75	74	0.2
Hot Spot 2	0.75	75	0.5	0.55	55	2.2	0.70	70	0.2

Table 8: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Mesh, oblivious						
Traffic	output			input fixed or random		
	load	xput	error	load	xput	error
Random	0.95	93	0.4	1.00	94	0.5
Bit reversal	1.00	61	0.2	1.00	61	0.2
Complement	0.45	44	0.5	0.45	45	0.5
Perfect shuffle	0.90	86	0.7	0.90	86	0.9
Transpose	1.00	72	0.2	1.00	72	0.2
Hot Spot 1	0.80	76	0.8	0.80	77	0.9
Hot Spot 2	0.75	71	1.2	0.75	71	1.2

Table 9: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Mesh, *-channels									
Traffic	output			input fixed			input random		
	load	xput	error	load	xput	error	load	xput	error
Random	1.00	93	0.4	1.00	92	0.5	0.95	92	0.4
Bit reversal	0.90	77	0.3	0.80	74	0.6	0.90	75	0.4
Complement	0.40	36	0.9	0.40	40	0.5	0.35	34	0.4
Perfect shuffle	0.90	89	0.4	0.85	84	0.6	0.90	89	0.5
Transpose	0.95	81	0.3	1.00	76	0.9	1.00	78	0.2
Hot Spot 1	0.95	87	0.8	0.95	87	0.8	0.90	89	0.6
Hot Spot 2	1.00	83	0.8	0.90	83	1.0	0.85	85	0.3

Table 10: Shows the normalized applied load (load) at which the maximum normalized throughput (xput) is achieved by the traffic pattern. The percent error (error) is also shown.

16x16 Mesh, minimal triplex									
Traffic	output			input fixed			input random		
	load	xput	error	load	xput	error	load	xput	error
Random	0.90	89	0.3	0.85	85	0.3	0.90	88	0.8
Bit reversal	0.70	65	0.4	0.65	61	0.4	0.70	65	0.7
Complement	0.35	34	0.5	0.40	40	0.5	0.35	32	2.8
Perfect shuffle	0.90	85	1.3	0.75	75	0.3	0.85	84	0.3
Transpose	1.00	82	0.2	0.90	76	0.6	1.00	78	0.2
Hot Spot 1	0.80	80	0.3	0.75	72	3.4	0.80	80	1.7
Hot Spot 2	0.80	80	0.3	0.75	71	2.7	0.80	80	0.3

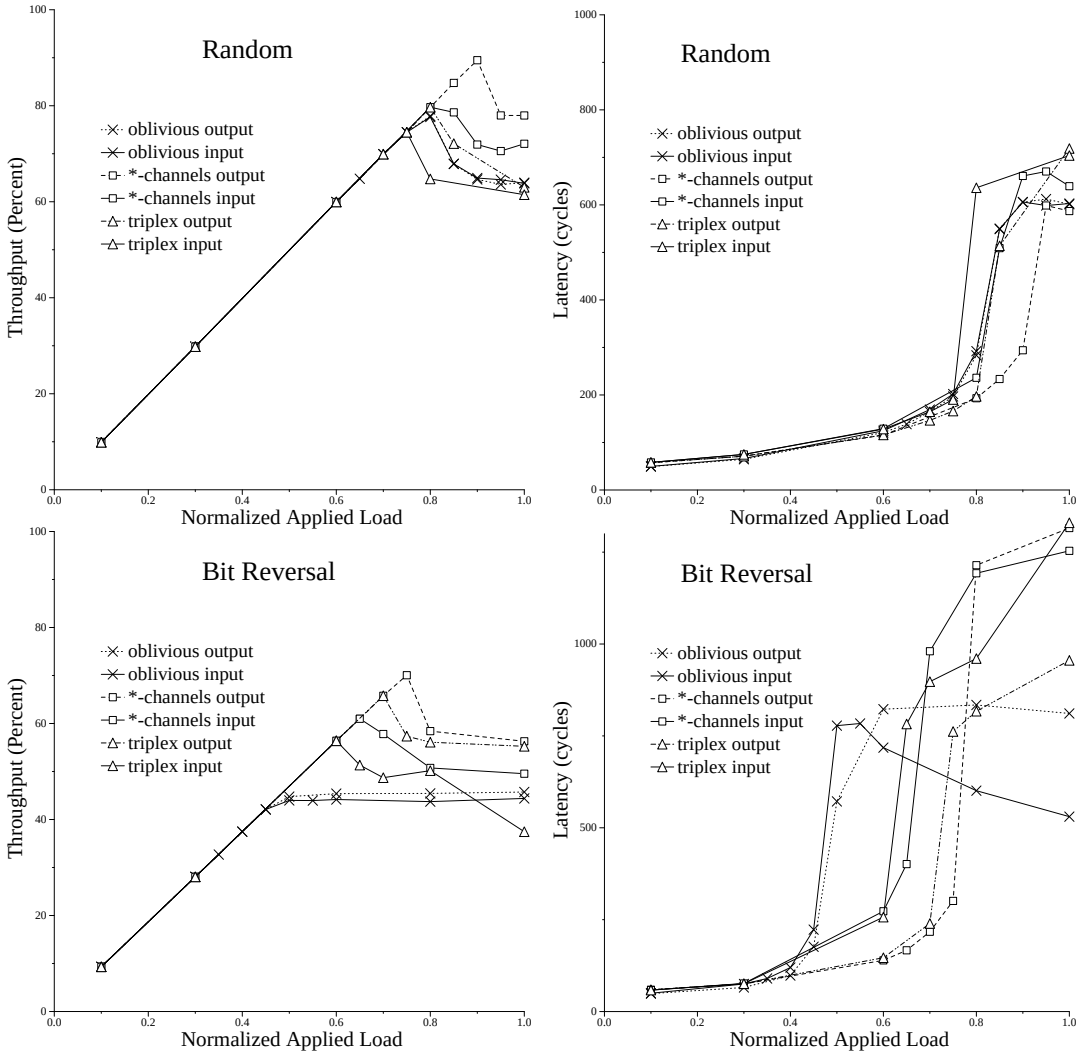

Figure 3: Throughput and latency on a 256-node 2D torus with fixed output buffer selection.

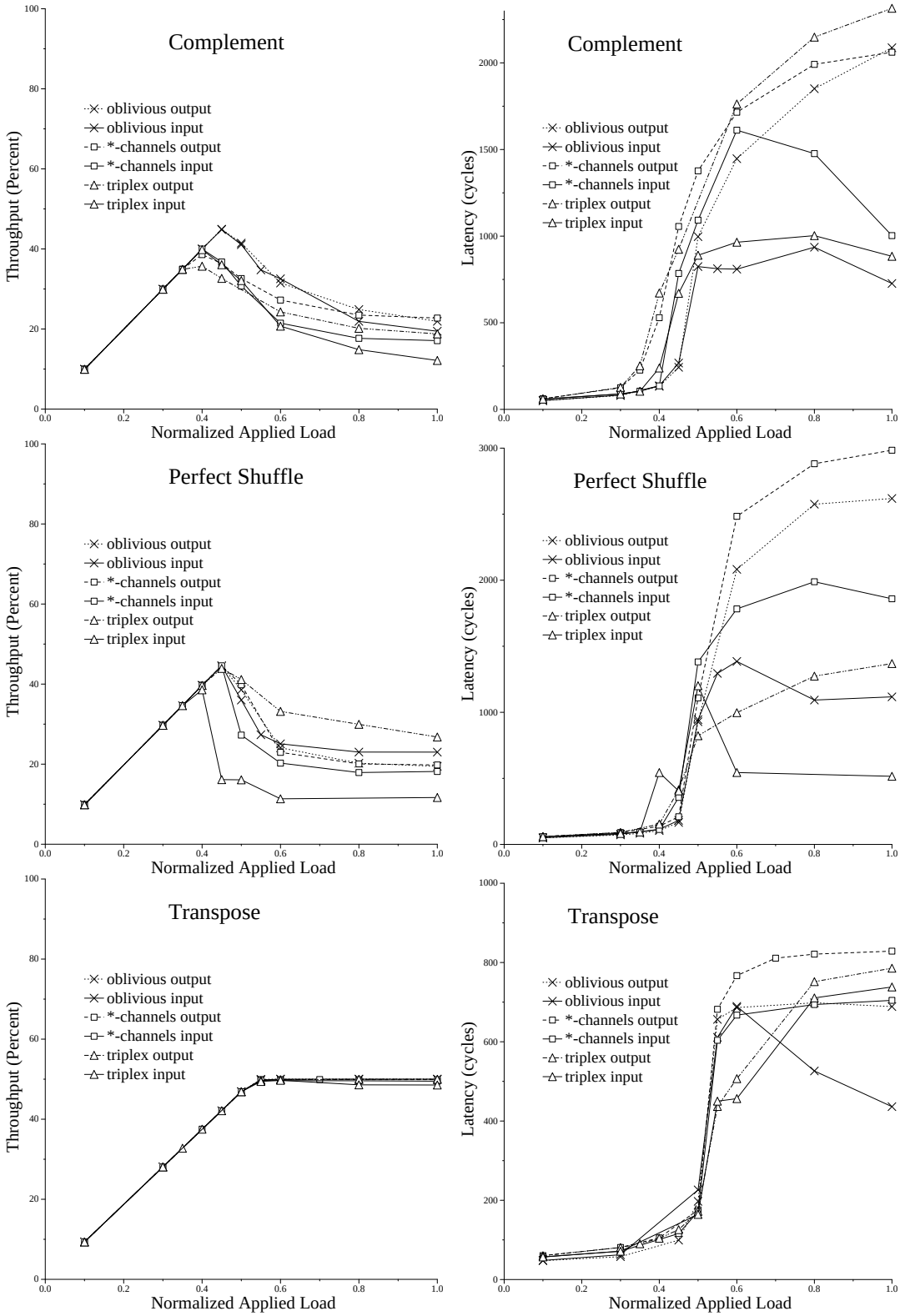

Figure 4: Throughput and latency on a 256-node 2D torus with fixed output buffer selection.

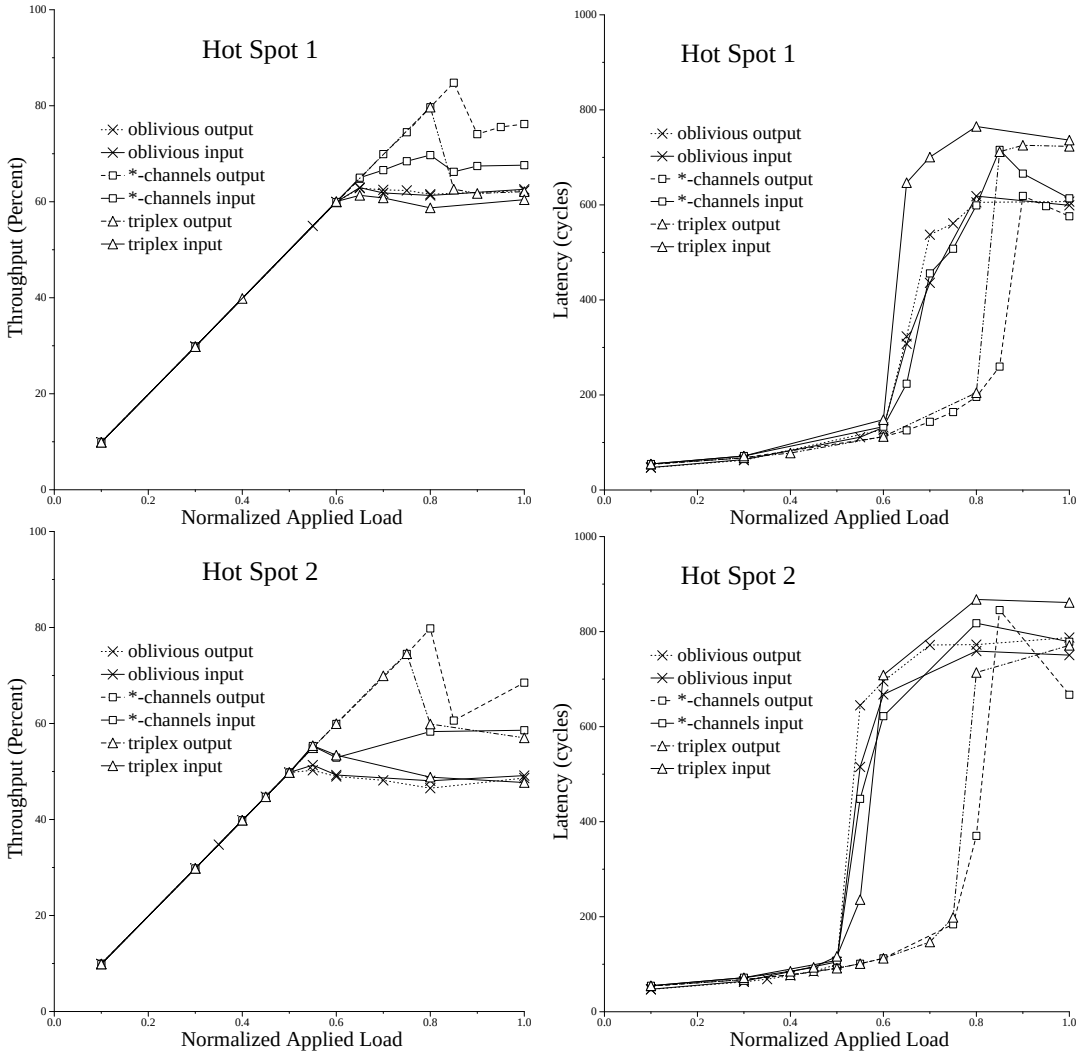

Figure 5: Throughput and latency on a 256-node 2D torus with fixed output buffer selection.

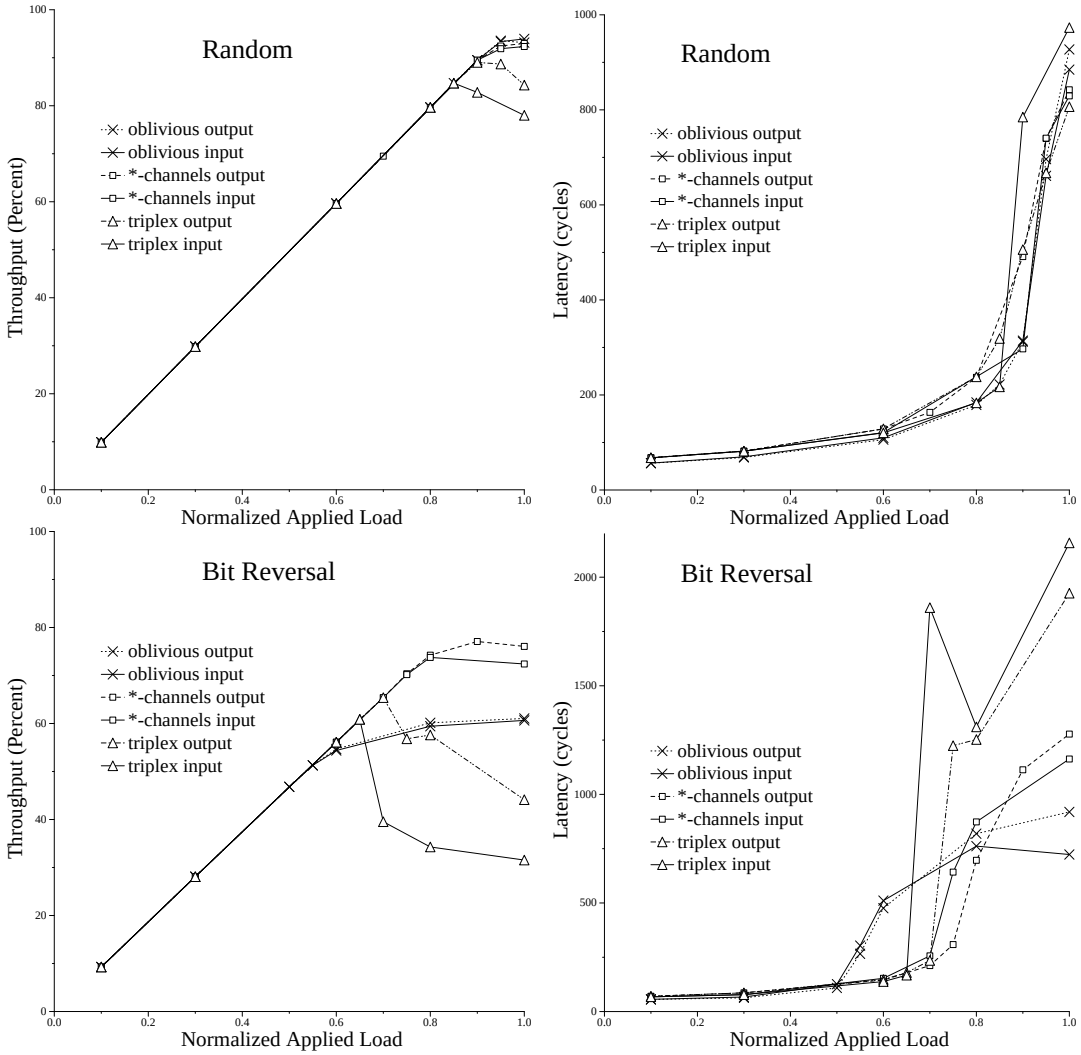

Figure 6: Throughput and latency on a 256-node 2D mesh with fixed output buffer selection.

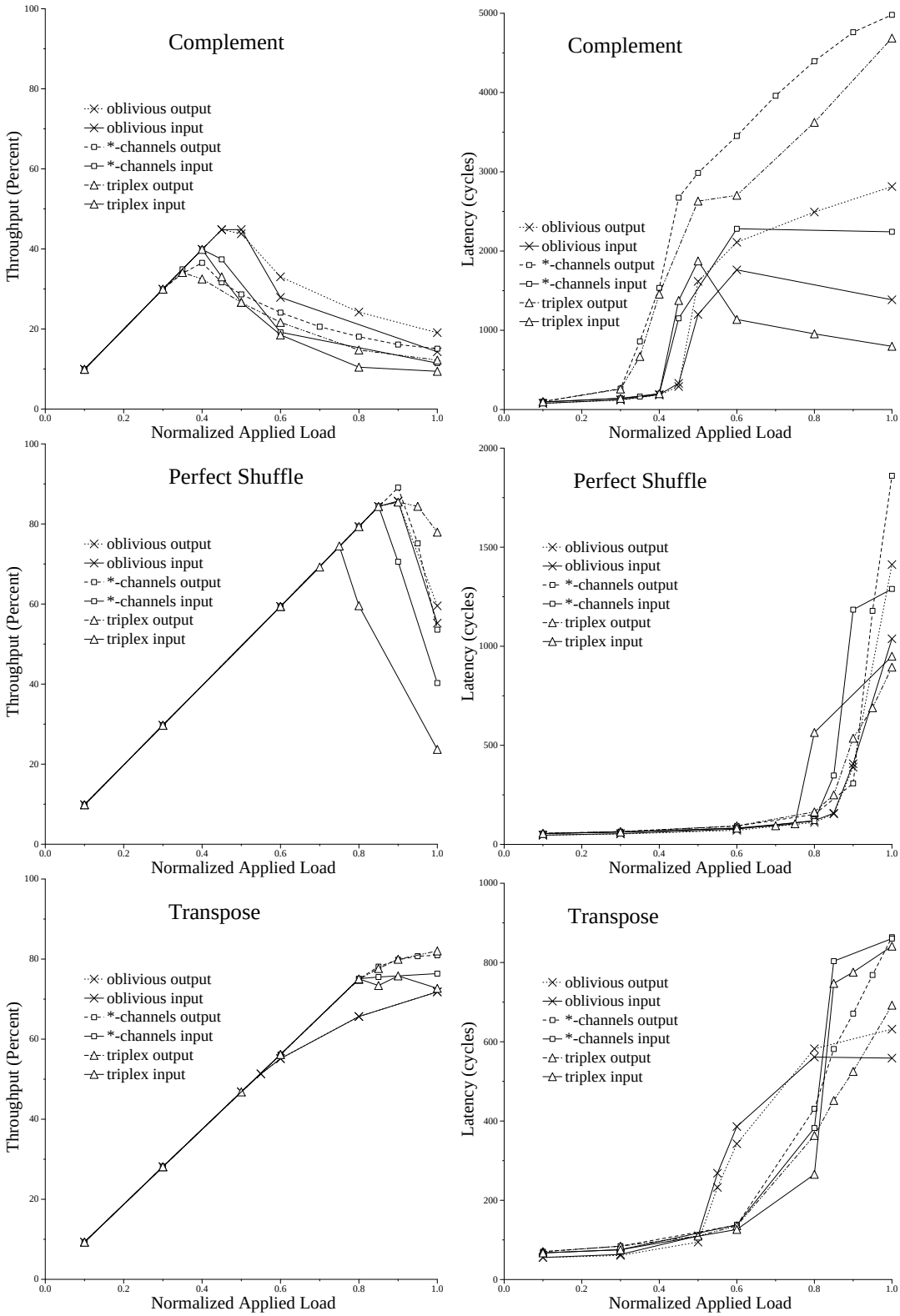

Figure 7: Throughput and latency on a 256-node 2D mesh with fixed output buffer selection.

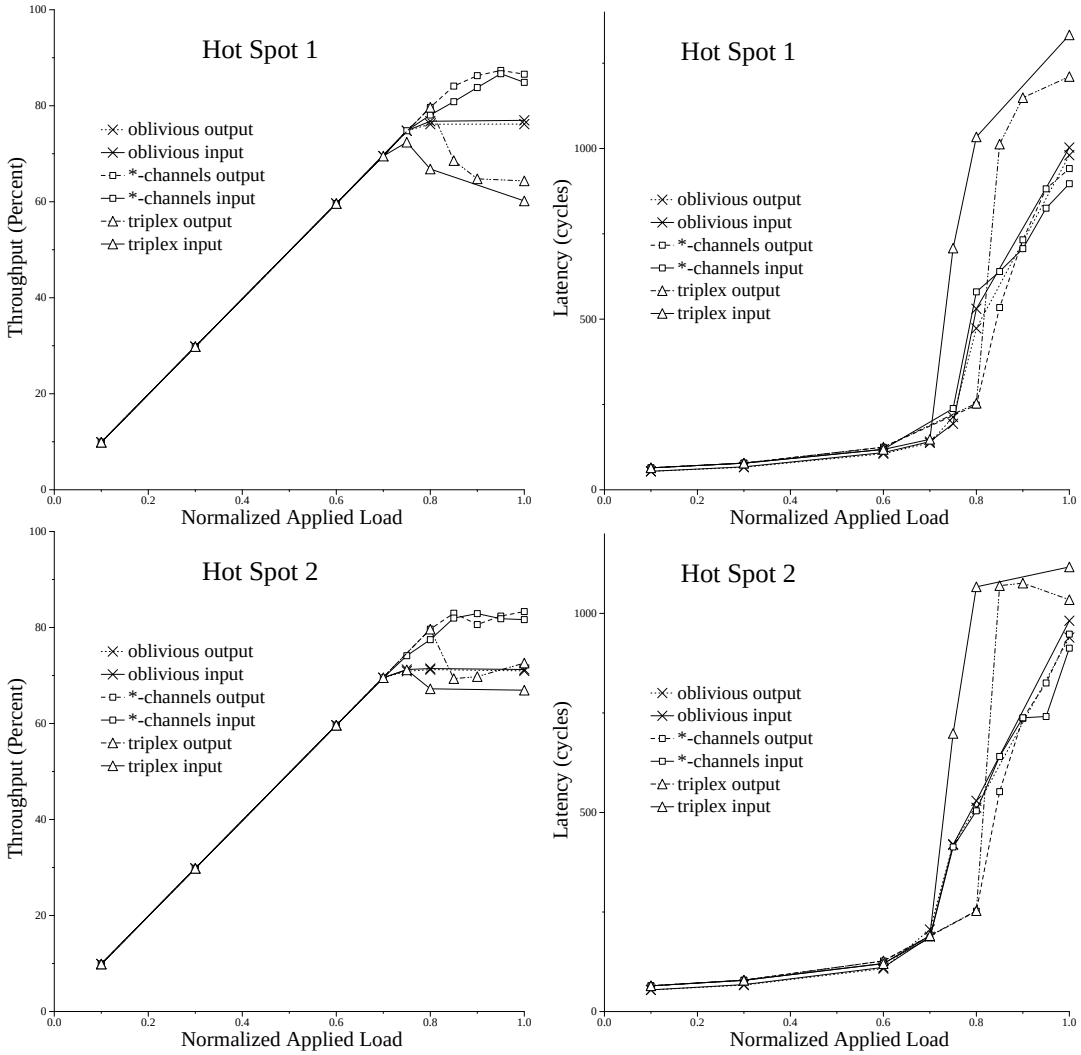

Figure 8: Throughput and latency on a 256-node 2D mesh with fixed output buffer selection.

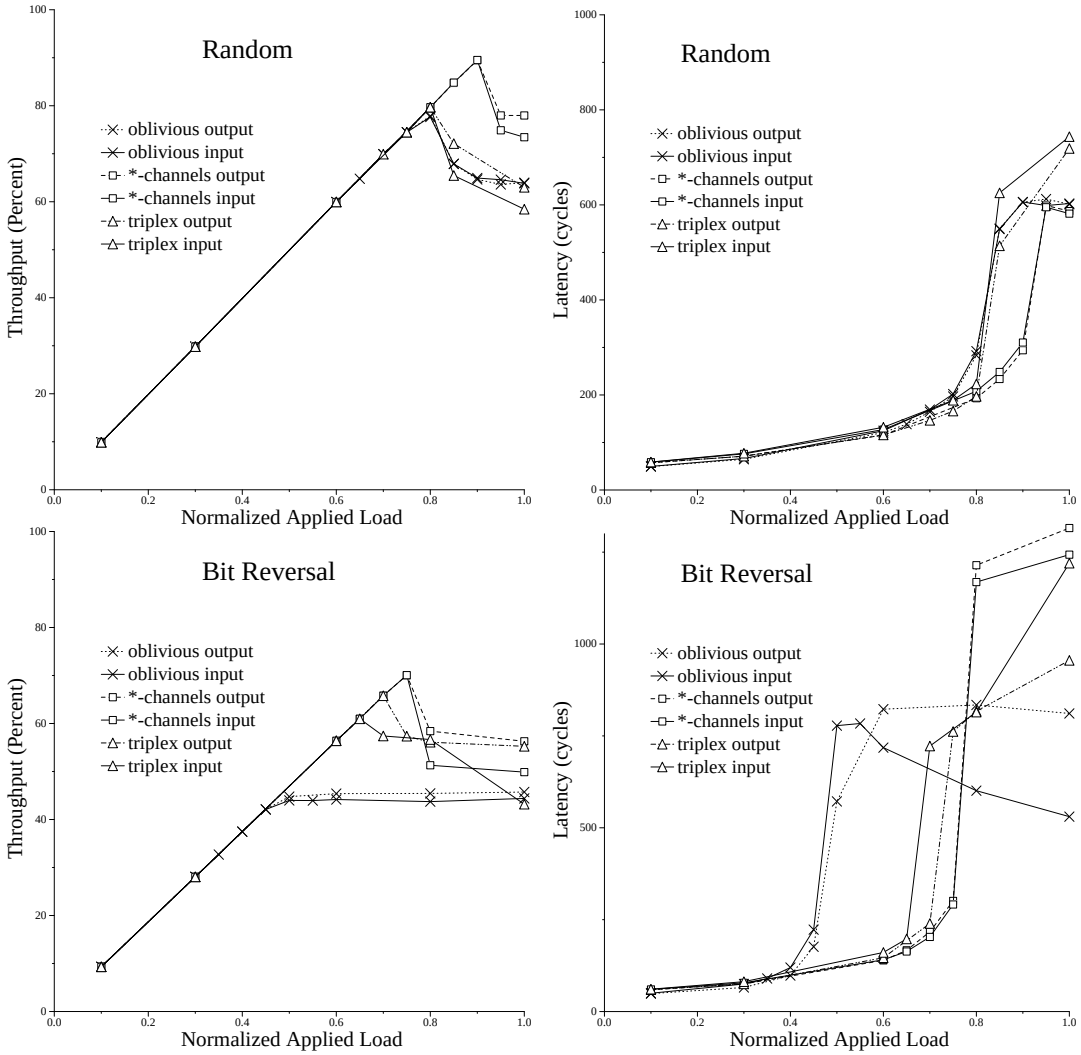

Figure 9: Throughput and latency on a 256-node torus with random output buffer selection.

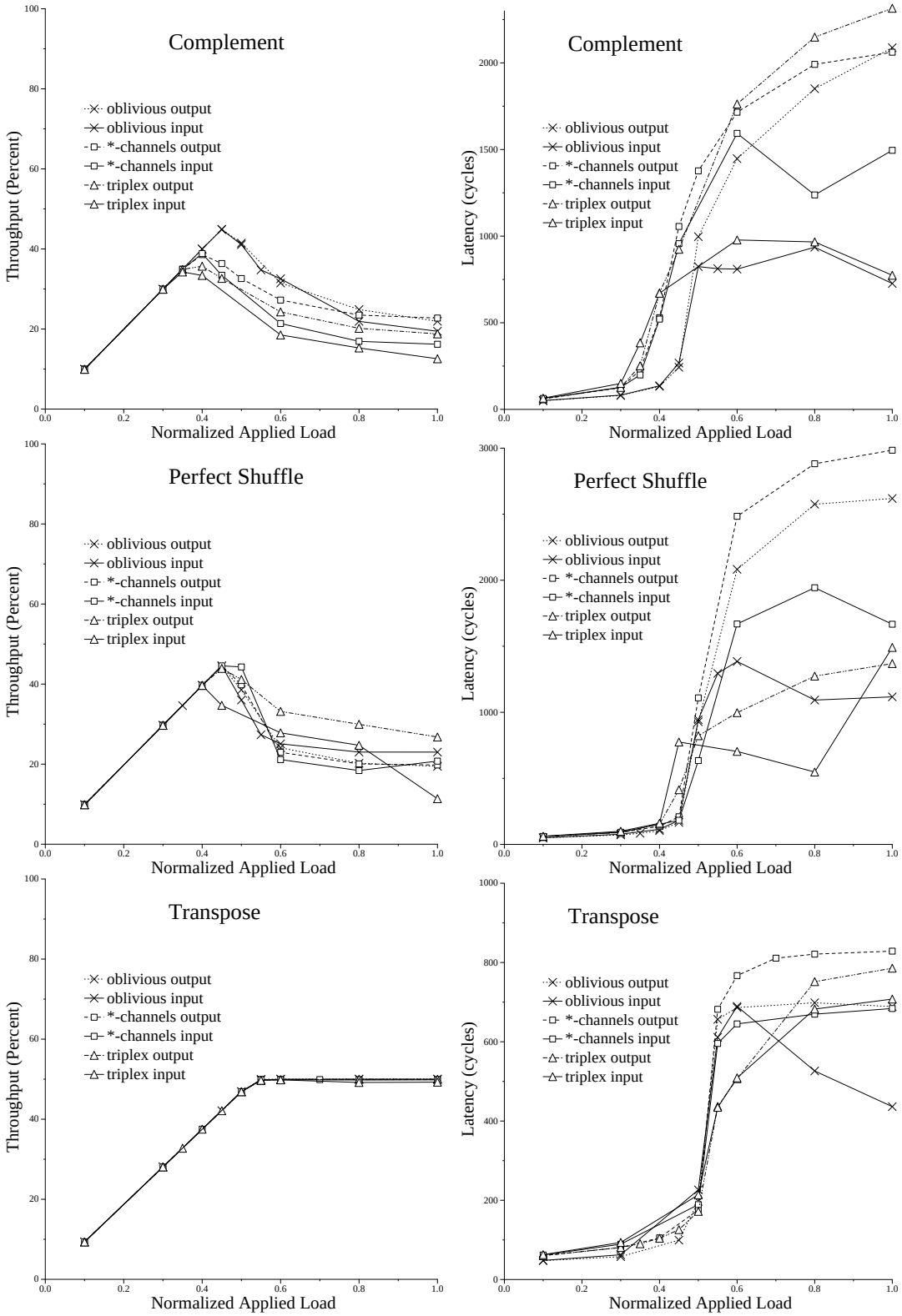

Figure 10: Throughput and latency on a 256-node torus with random output buffer selection.

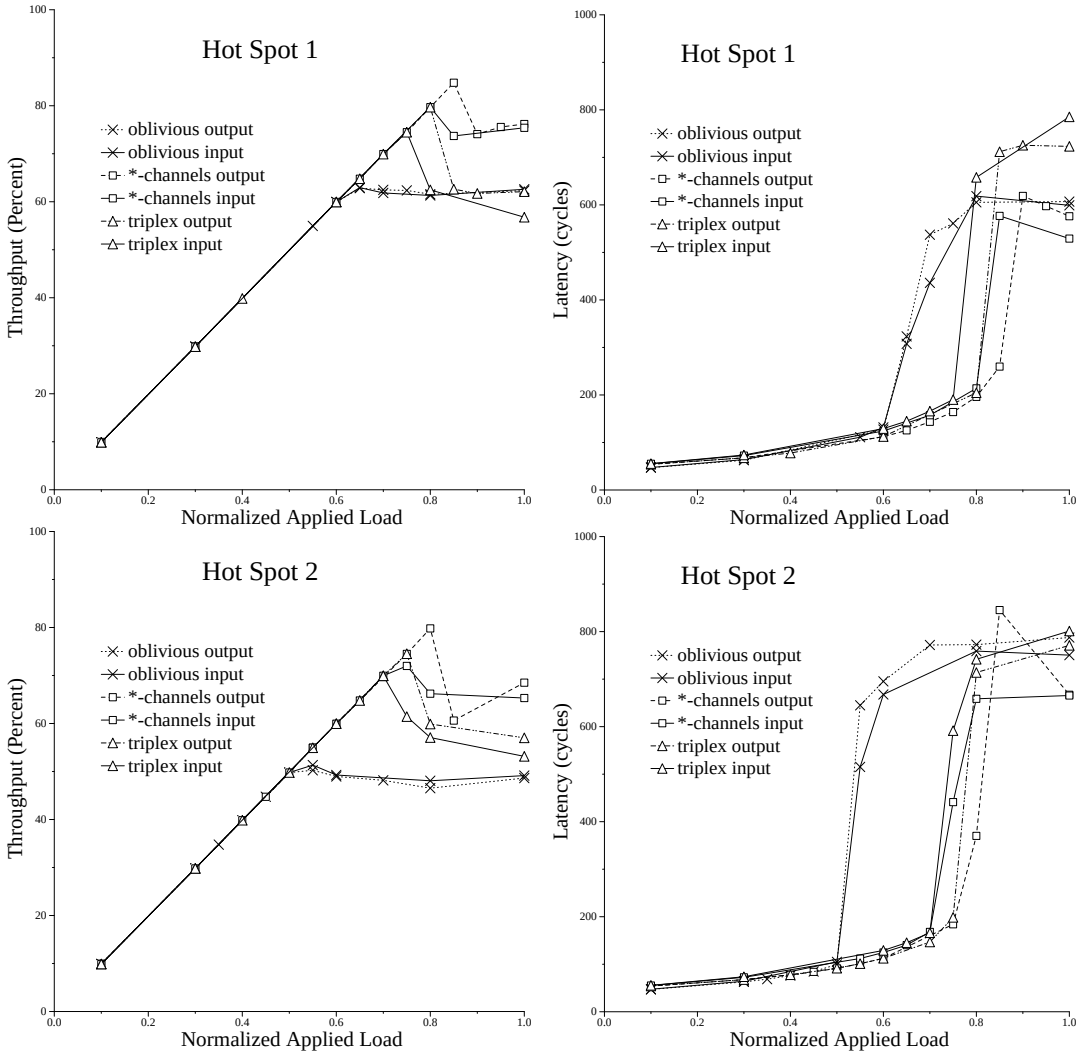

Figure 11: Throughput and latency on a 256-node torus with random output buffer selection.

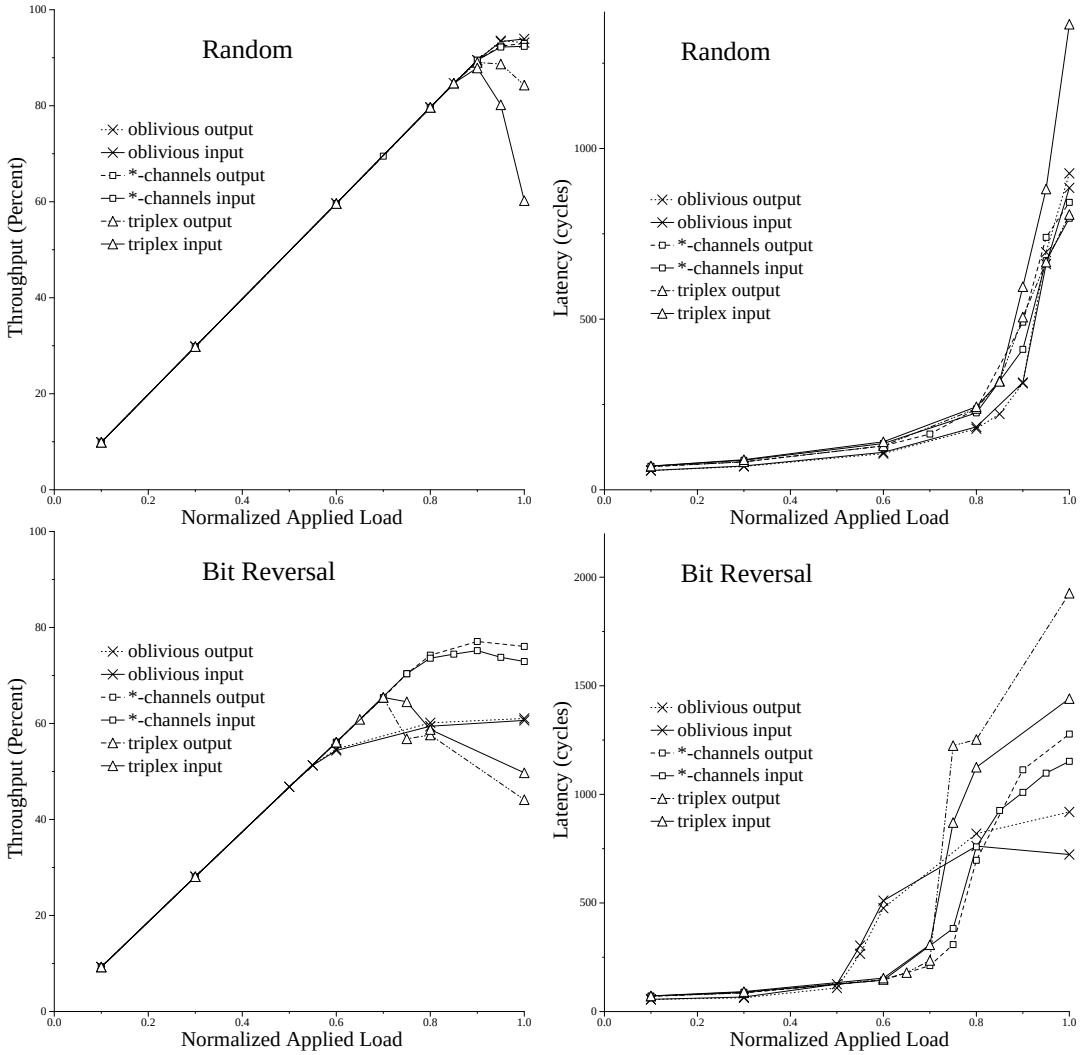

Figure 12: Throughput and latency on a 256-node mesh with random output buffer selection.

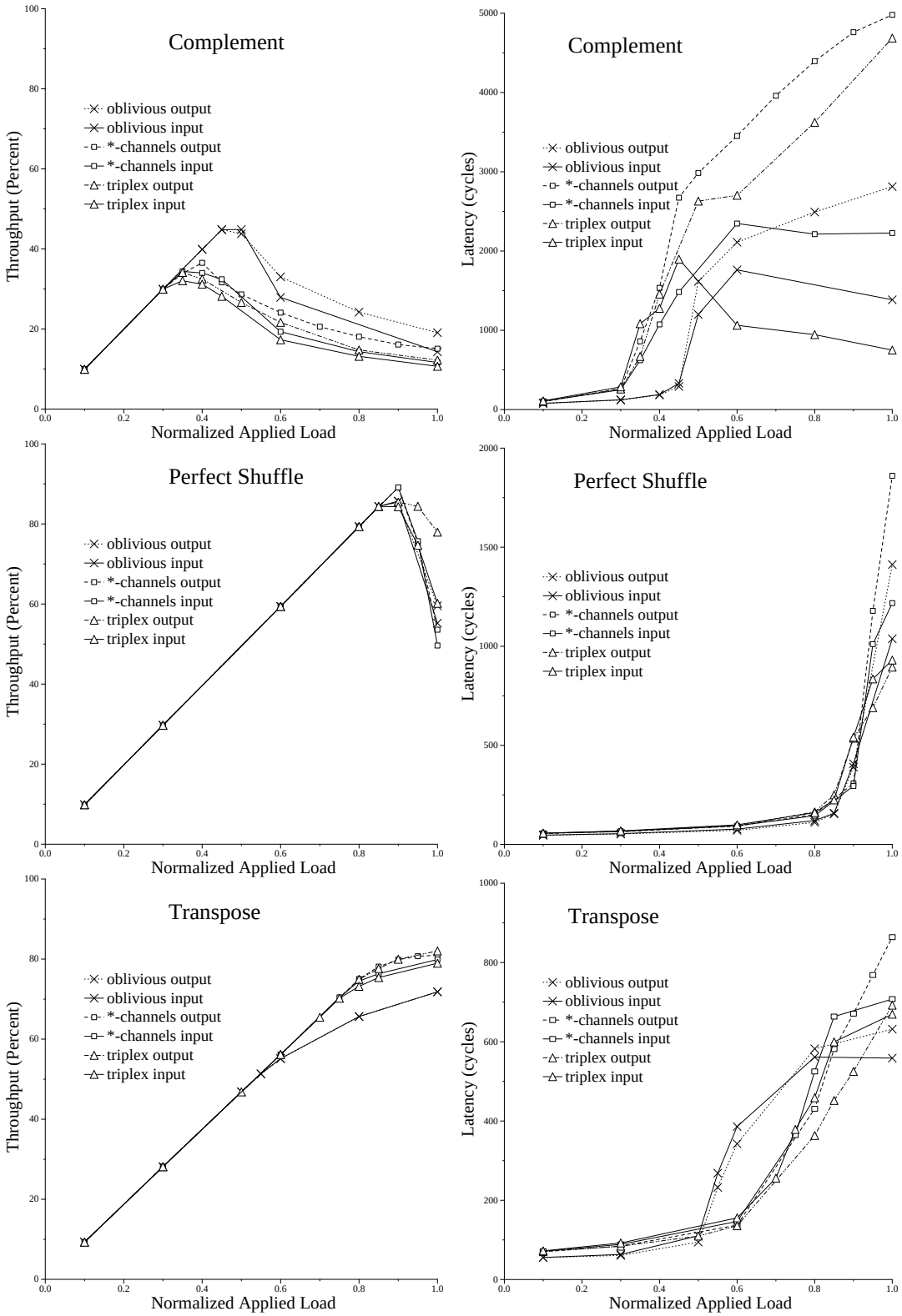

Figure 13: Throughput and latency on a 256-node mesh with random output buffer selection.

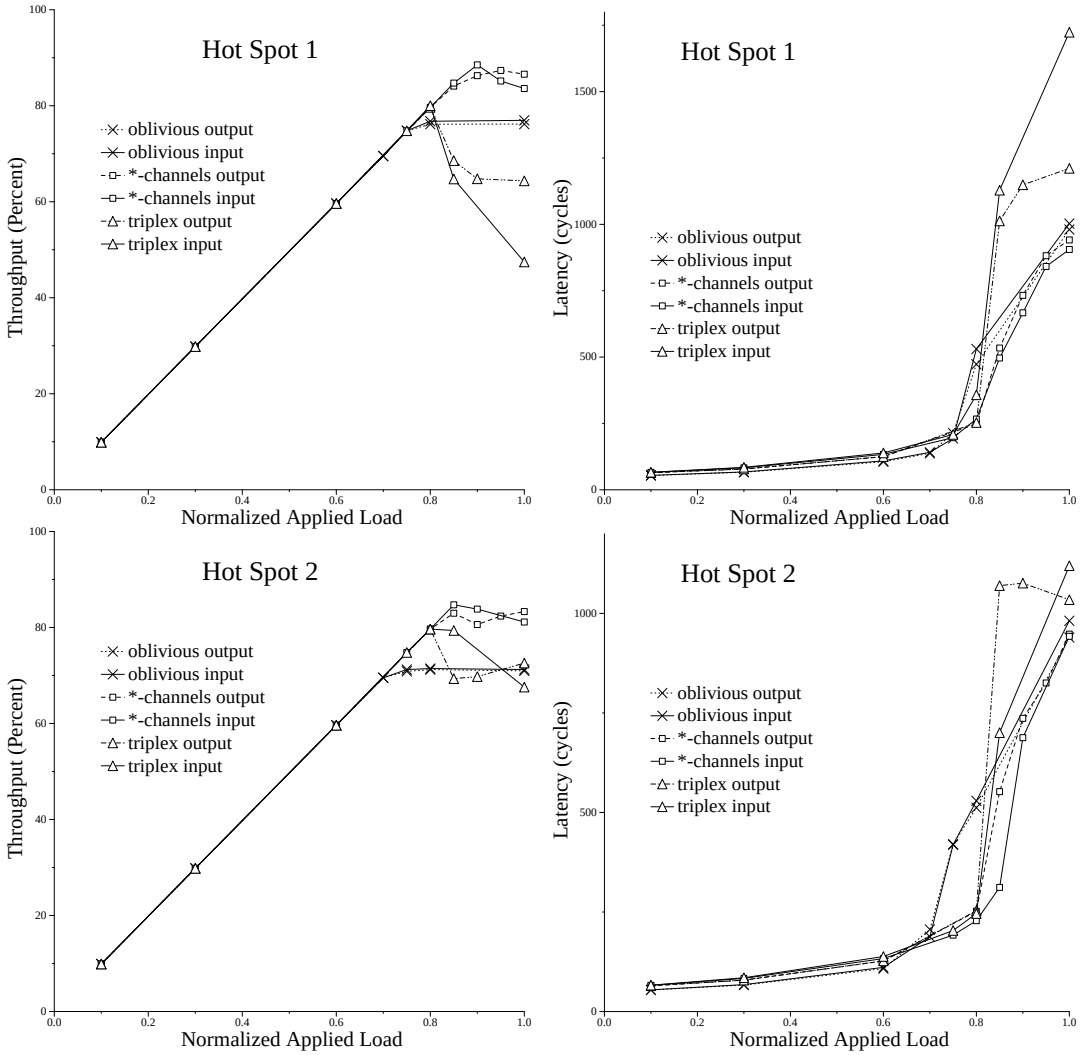

Figure 14: Throughput and latency on a 256-node mesh with random output buffer selection.

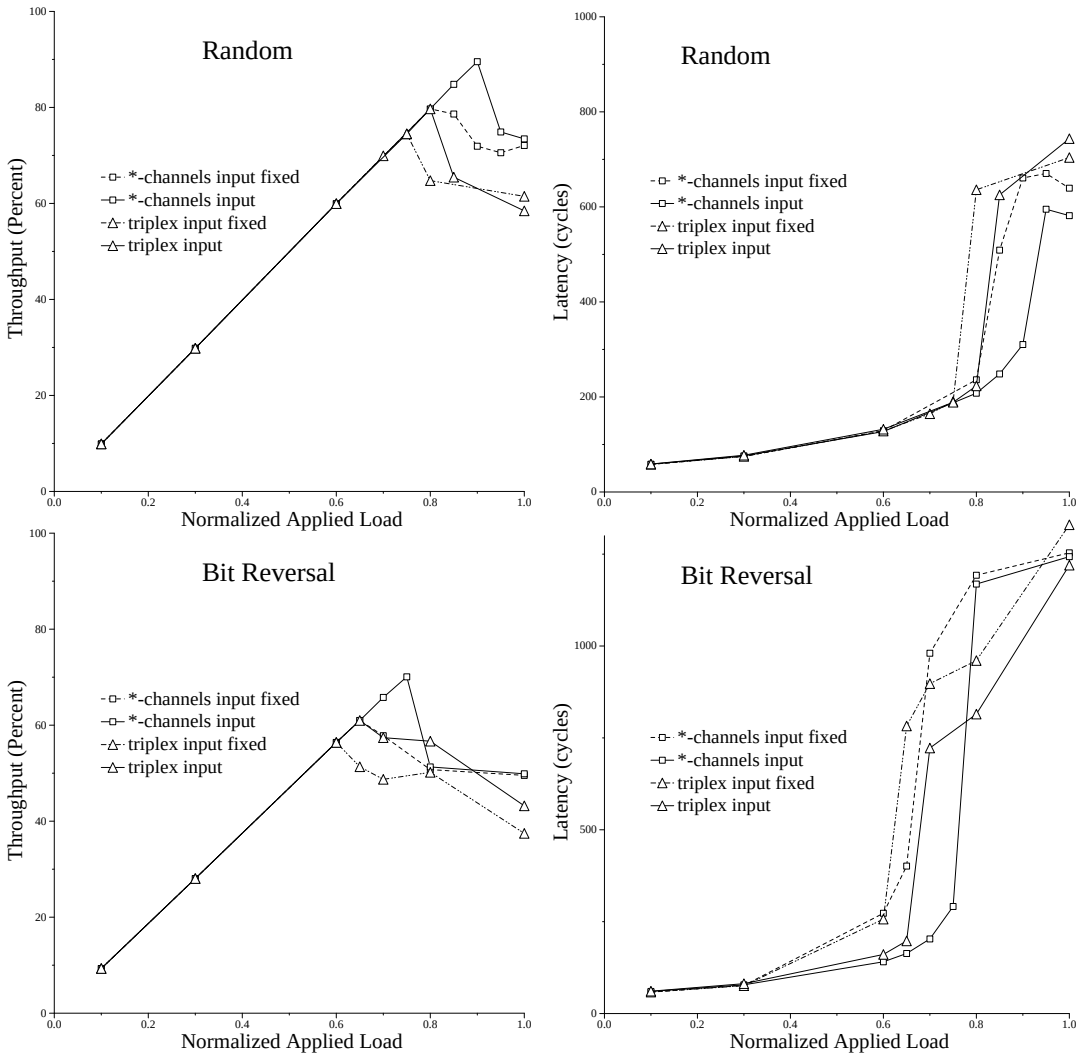

Figure 15: Throughput and latency on a 256-node torus comparing input driven routing with fixed and random selection.

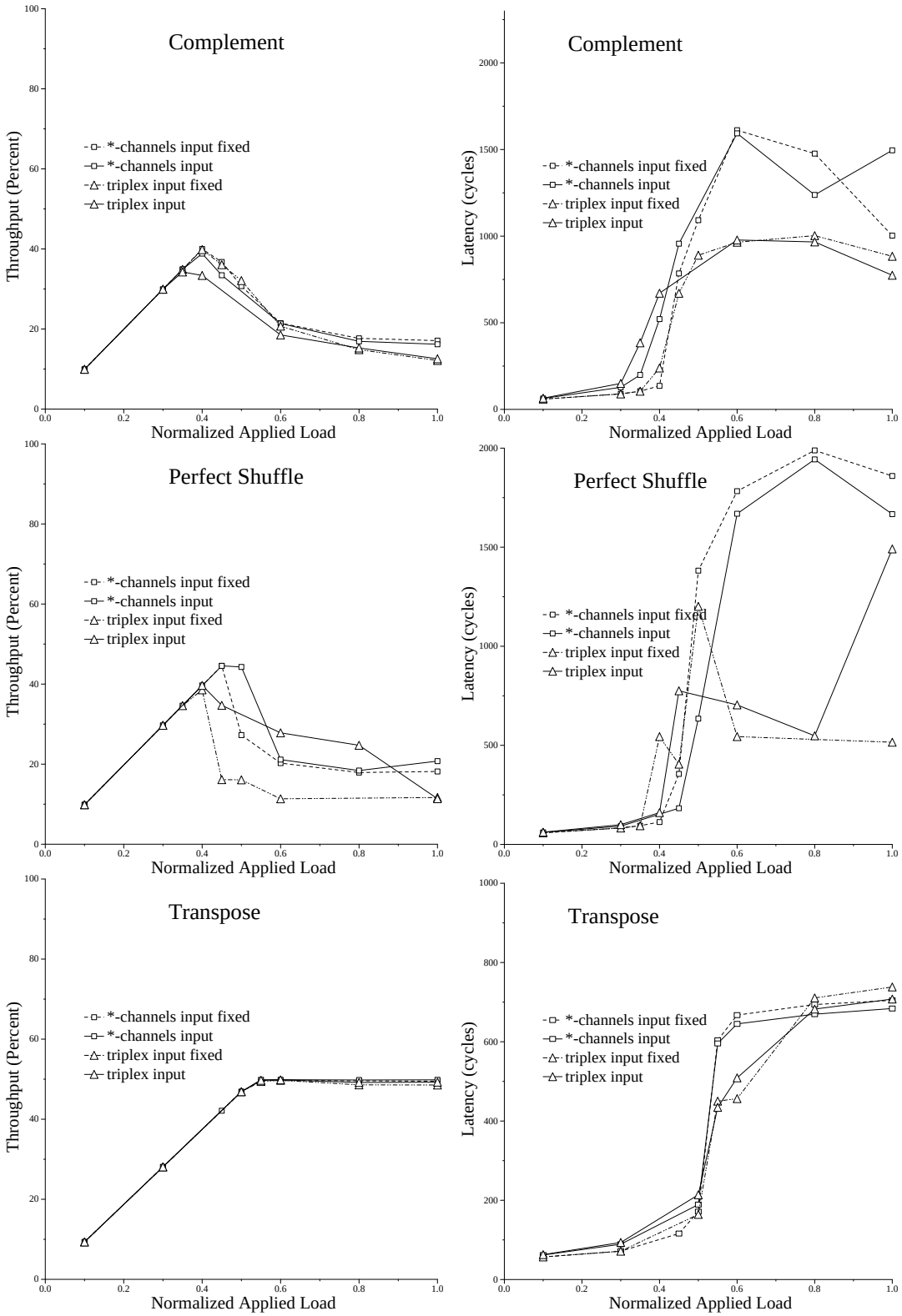

Figure 16: Throughput and latency on a 256-node torus comparing input driven routing with fixed and random selection.

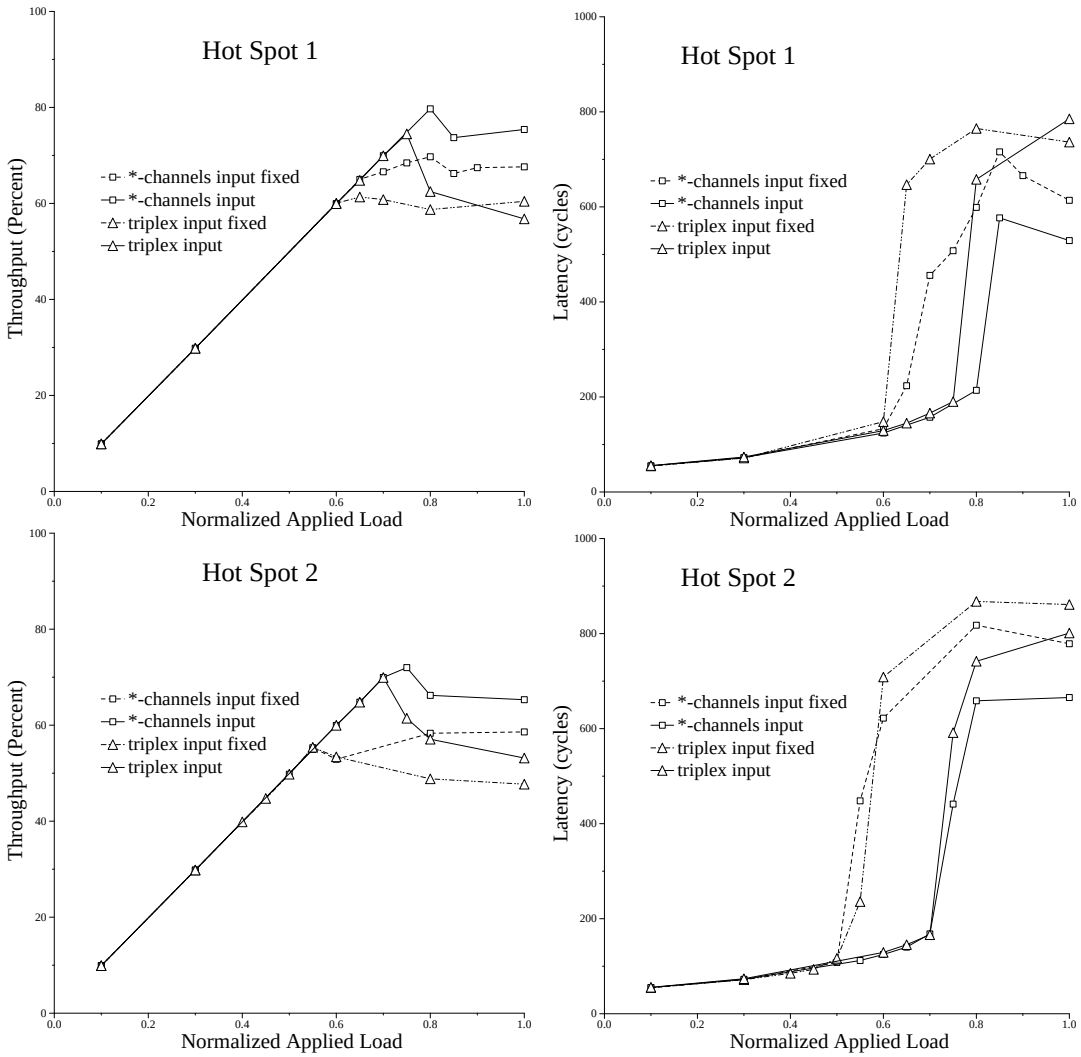

Figure 17: Throughput and latency on a 256-node torus comparing input driven routing with fixed and random selection.

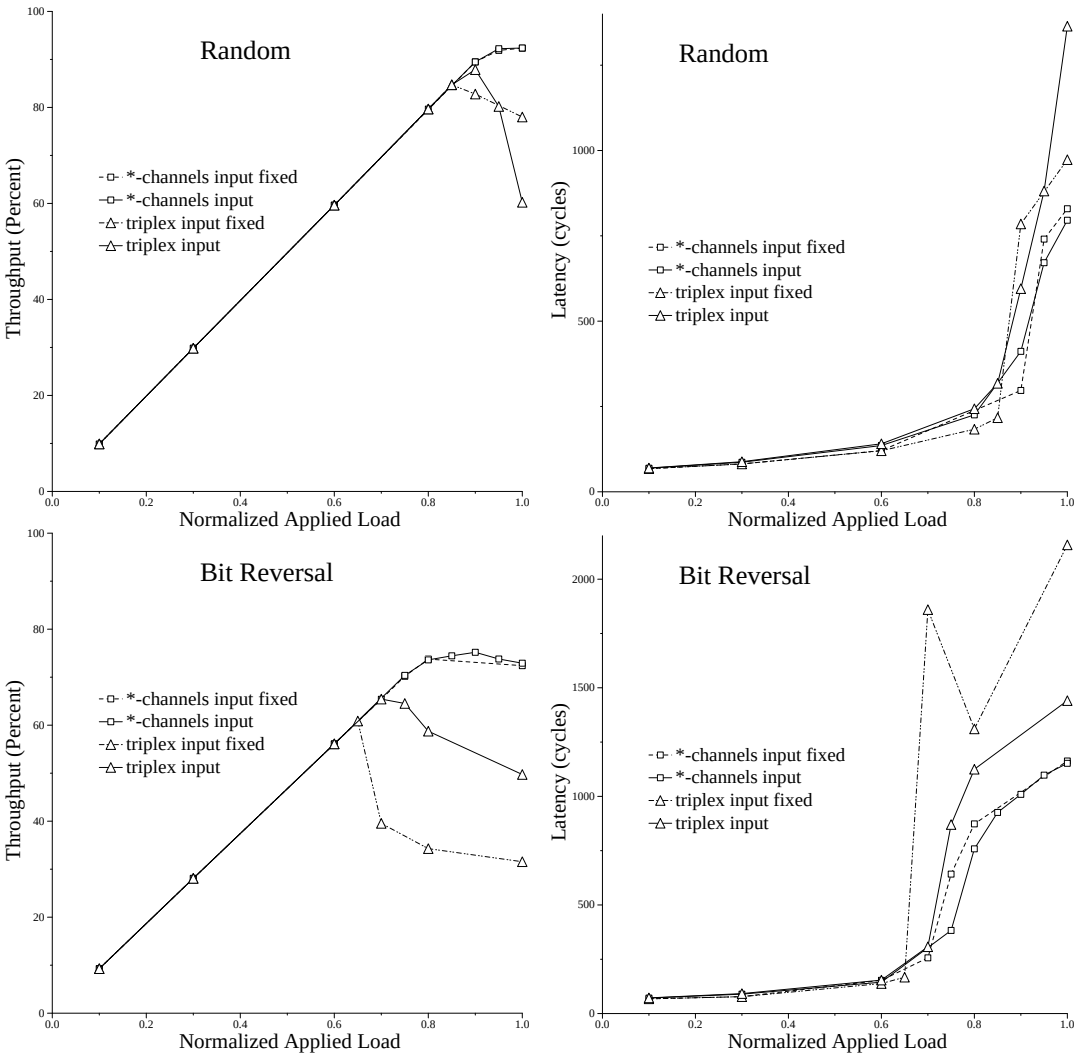

Figure 18: Throughput and latency on a 256-node mesh comparing input driven routing with fixed and random selection.

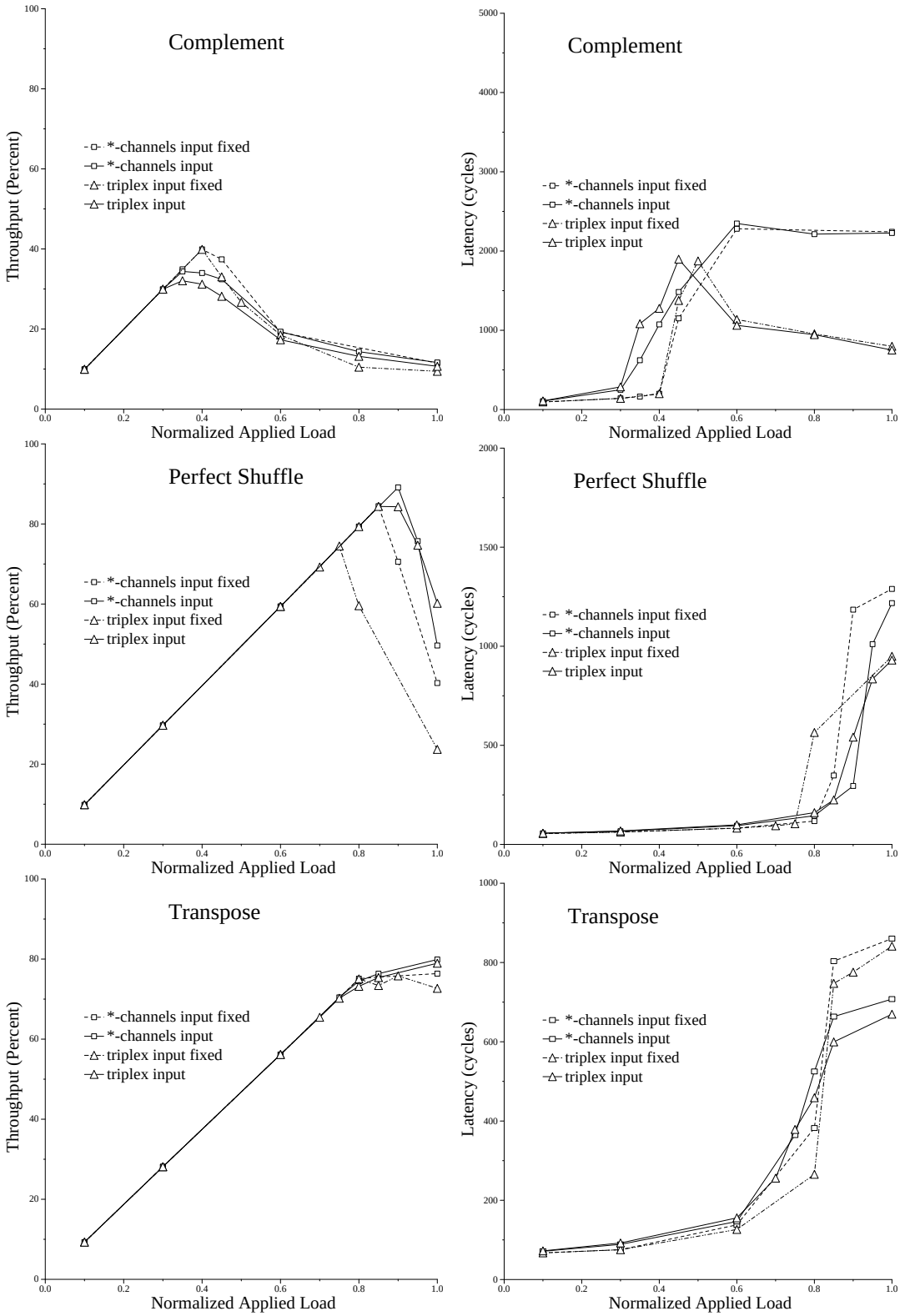

Figure 19: Throughput and latency on a 256-node mesh comparing input driven routing with fixed and random selection.

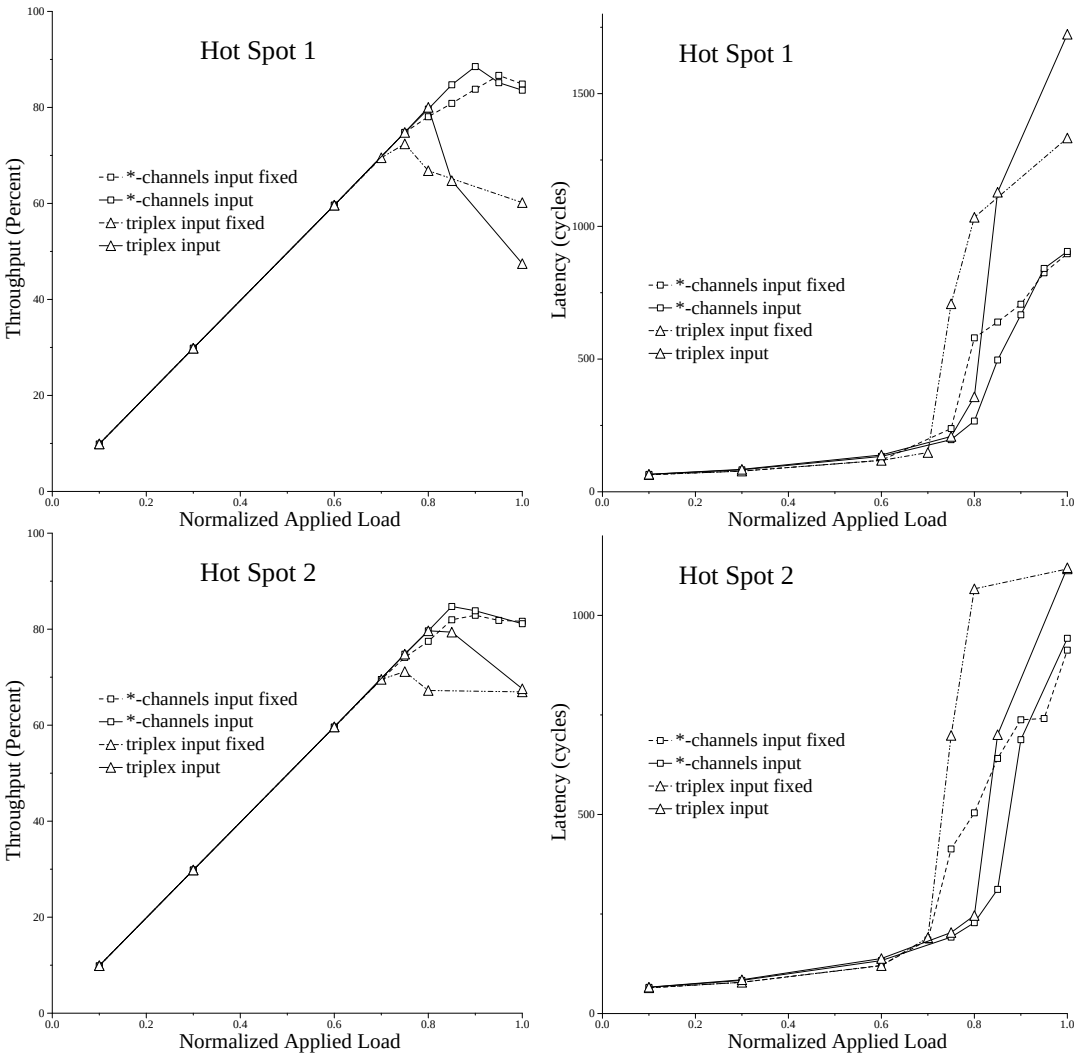

Figure 20: Throughput and latency on a 256-node mesh comparing input driven routing with fixed and random selection.