Triplex Router: A Versatile Torus Routing Algorithm

Melanie L. Fulgham
Lawrence Snyder

Technical Report UW-CSE-96-01-11

Department of Computer Science and Engineering
University of Washington

Department of Computer Science and Engineering, Box 352350
University of Washington, Seattle, WA 98195-2350 USA

Triplex Router: A Versatile Torus Routing Algorithm[†]

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

Abstract

Parallel applications exhibit a variety of communication styles from small, frequent messages to a few, very large messages containing data sets. The ability of the network to satisfy the application's communication requirements depends on the routing class implemented in the hardware router. For example, in-order message delivery is provided by an oblivious class, but not an adaptive one; while high sustainable throughput for large amounts of data is best satisfied by a fully adaptive router, rather than an oblivious router. Current routers provide a single routing class. If the class is oblivious, the network provides in-order delivery and low latency under lightly loaded conditions, but suffers from poor performance in highly congested situations. Minimal fully adaptive routers provide higher throughput than oblivious routers in congestion, but performance degrades dramatically in many cases if the traffic burst is too large. Non-minimal routers provide high throughput and low latency in congested networks and for large bursts of traffic, but come with the cost of livelock protection.

In this paper we present the Triplex router, a versatile, multi-class routing algorithm which allows dynamic selection of oblivious, minimal fully adaptive, and non-minimal fully adaptive routing classes. Flow control is either by wormhole routing or packet-switched techniques. Furthermore, the algorithm is of independent interest since we establish for the first time, deadlock avoidance for a non-minimal fully adaptive wormhole algorithm on the torus. Simulations on a 256-node two dimensional torus compare Triplex with the Dally-Seitz oblivious router, Duato, and the Chaos router. These routers represent competitive oblivious, minimal fully adaptive, and non-minimal fully adaptive classes; the same classes offered by Triplex. Results show Triplex does not suffer significant performance degradation for oblivious and minimal classes despite providing all three routing classes, instead of the usual single class offering. The performance of the non-minimal class, however, may not justify its inclusion in a Triplex router implementation.

[†]This work supported in part by NSF Grant MIP-9213469 and by an ARPA Graduate Research Fellowship.

1. Introduction

Communication performance continues to be a significant problem in parallel computers. There are two components of communication performance: the overhead required to send a message and the latency of the message in the interconnection network. Until recently, the large software overhead required to send a message through the network has overshadowed the need for low latency routing algorithms, the rules which specify a message path. As communication overhead declines [vEDCGS92, Dal90, BLA+94, MBES94], however, the routing algorithm becomes more important to communication performance. Research in reducing communication overhead has resulted in prototype communication systems with network time accounting for over 50% of the typical communication time, even when no contention occurs [CKP+93]. In the future, routing latency may even dominate communication costs. In this paper, we consider low latency routing algorithms for arbitrary torus networks. The torus is the topology for the Cray T3D and the underlying structure of the Tera parallel computer.

Low latency, however, is only one of the concerns. Parallel applications have a widely varying set of communication styles which place diverse and sometimes conflicting demands on the network. For example, some tasks require in-order message delivery, others low latency in the presence of congestion, and still others high throughput for large amounts of data. Each of these requirements is best satisfied by a different routing class.

If the single class is oblivious[1], the routing algorithm can deliver messages in-order. In-order delivery is useful for synchronization, helps provide determinism in parallel computation, and is usually assumed by software designers. Furthermore, an oblivious router is simple and performs well under lightly loaded conditions where network latency is the dominant factor. However because it lacks adaptivity, an oblivious router performs poorly in a congested network, and fails entirely when there are faults in the network.

A minimal fully adaptive router is more complex; and despite having adaptivity limited to all shortest paths, handles small amounts of congestion very well. Unlike the oblivious class, it does not provide in-order delivery since adaptive routing allows one message to overtake another. Under severe congestion, as from a large communication burst, the throughput of a minimal adaptive algorithm usually degrades dramatically. Nevertheless, a minimal adaptive router is sometimes favored over a non-minimal adaptive router because the latter requires livelock protection[2].

A dedicated non-minimal, fully adaptive routing class[3] delivers the low latency and high throughput performance needed for large bursts of traffic; but this performance comes with a cost. In-order delivery is not supported; there is additional complexity from preventing livelock in the network; and occasionally, severe congestion causes latency to increase dramatically because too many non-minimal routes are selected.

Clearly, there is a complexity-performance tradeoff for each of these classes. If the oblivious router always provided the best performance, the choice of routing classes would be easy. If this performance is inadequate, a fully adaptive router provides superior results, but

[1] The path of a message is fixed by its source and destination.

[2] Livelock occurs when a message circulates in the network and is never delivered.

[3] Such a class prefers any shortest path, but is permitted to take a longer path when necessary.

sacrifices simplicity and in-order message delivery. At present, there is no router that is best suited for all kinds of workloads. This suggests that routers that support several classes at once or that change classes dynamically may meet the requirements of a more diverse set of workloads. Until now, traditional routers have implemented a single routing class in the network hardware.

In this paper we present the Triplex router, a versatile, multi-class router for torus networks. The router provides oblivious, minimal fully adaptive, and non-minimal fully adaptive routing classes for both wormhole or packet-switched flow control, including virtual cut-through [KK79] and store-and-forward techniques. The choice of class may be specified at system boot-up, dynamically, or individually by message. Dynamic class selection may be useful when compile time information is available [Fel93], enabling the system to select the best class for the expected traffic. Individual selection is useful, for example, when some messages require in-order delivery, while other messages prefer the increased flexibility of adaptive routing.

It is obvious that a multi-class router is unlikely to be superior to the best router for any given class, since specialization usually outperforms generality. The question is, what price must be paid for the flexibility of a multi-class router? This will be determined by comparing the throughput and latency of Triplex to those of the best router for each class. We will see that with the oblivious and minimal adaptive classes, Triplex is competitive with the best. For non-minimal adaptive routing, however, Triplex's generality is limiting. Thus, an open question will remain. Is there a no-penalty multi-class router?

Triplex is also of independent interest, since it is the first non-minimal fully adaptive deadlock avoidance wormhole algorithm for the torus.

The remainder of the paper is organized as follows. The routing framework is described in Section 2. The Triplex algorithm is specified in Section 3. Section 4 experimentally compares Triplex with each of the following single class routers: the Dally-Seitz oblivious router, Duato, and the Chaos router. Related work follows in Section 5, and conclusions are presented in Section 6.

2. Framework

This section describes the network assumptions and routing terminology used in this paper. Each node in the network has a finite number of buffers with two distinguished buffers: the injection buffer, where messages are injected into the network and the delivery buffer, where messages are removed from the network within a finite amount of time. Each node is connected to a set of neighboring nodes by physical channels which allow communication in both directions. For wormhole routing, the standard unidirectional (simplex) channels are assumed. For packet routing either unidirectional channels or a bidirectional (half-duplex) channel shared between the two directions can be used. In this framework, a virtual channel[4] will often be referred to by its corresponding buffers. Arbitration among buffers is done in a manner which prevents starvation. We assume for the wormhole case that at most a single message can reside in a buffer at any time. For packet routing, we assume a buffer can hold

[4] A technique which uses several sets of buffers to give the appearance of multiple physical channels.

an entire packet exactly. An extension to larger buffers is straight forward.

The routing algorithm makes local decisions only. The algorithm consists of two functions: the *routing* function, which computes the possible set of buffers that may be used by a message, and the *selection* function, which chooses one of the buffers from the set output by the routing function. For example, the selection function can make decisions based on the presence of other messages in neighboring buffers. A *waiting buffer* is a buffer a message can wait to acquire when all other buffers specified by the routing function cannot be selected. A routing algorithm is *wait-connected* if a message always has at least one waiting buffer [SJ96].

3. The Triplex Algorithm

Triplex is a non-minimal fully-adaptive deadlock-free packet-switched or wormhole routing algorithm for n-dimensional torus networks. Our algorithm is the first non-minimal fully adaptive, deadlock-free wormhole routing algorithm for n-dimensional tori using deadlock avoidance[5]. Although the algorithm is non-minimal, no message is forced to take a non-minimal route. This allows a message to choose the kind of routing it prefers such as oblivious, minimal, or non-minimal classes. The algorithm gains adaptivity by avoiding the traditional requirement of a connected set of channels with static acyclic channel dependencies. Depending upon the implementation, the algorithm is also deterministically or probabilistically livelock-free[6]. For presentation clarity the mesh algorithm is presented first, and the torus algorithm follows. There are two versions of the algorithm: one for wormhole routing and a slightly more flexible and less complex one for packet-switched routing. The mesh algorithm uses two virtual channels per channel, while the torus algorithm uses three virtual channels per channel.

Let *DO* be the dimension order Dally-Seitz oblivious, deadlock-free wormhole routing algorithm [DS87]. The *DO* algorithm routes a message from the lowest dimension to the highest dimension, where the direction in each dimension is chosen to make the message route minimal. This algorithm is wait-connected and deadlock-free. Although the specific *DO* rules differ slightly for the mesh and the torus, the particular details are not relevant. Hence, the mesh algorithm will not be distinguished from the torus algorithm except by context. The *DO* function will be used as a subroutine by the Triplex routing algorithm.

The virtual channels are partitioned into two classes. The *restricted* class refers to the set of virtual channels used by the *DO* subroutine, two virtual channels per channel for the torus and one for the mesh, while the *unrestricted* class contains the remaining virtual channel for a total of three virtual channels for the torus and two for the mesh.

A buffer is *unrestricted* if its corresponding virtual channel is unrestricted, is *restricted* if its corresponding virtual channel is restricted, and is called a *wrap* buffer if its corresponding virtual channel is a wrap channel[7, 8]. For wormhole routing, a buffer representing a virtual

[5]Deadlock-freedom achieved by avoiding deadlocks.

[6]The probability a message is still circulating in the network goes to zero as time goes to infinity.

[7]A wrap channel in dimension i is simply a single distinguished channel in dimension i, though by convention it is often the channel between the last and first node of dimension i.

[8]For packet routing, only the output buffer corresponding to the wrap virtual channel is considered a wrap buffer.

channel is considered *empty* when both the input and output buffers that compose the virtual channel are empty. This requires state information from a neighboring node. For packet routing, a non-full buffer is considered *empty* since once a packet header progresses to a buffer, it is guaranteed to be completely accepted into this buffer, even if the packet becomes blocked. Messages may only move to empty buffers.

There is a *direct waiting dependence* from buffer a to buffer b if a message can use buffer a and wait for waiting buffer b. By transitivity, there is also a *waiting dependence* between a and b if there is a sequence of buffers $(a = b_1, b_2, \ldots, b_s = b)$ such that there is a message m_i that uses buffer b_i and waits for waiting buffer b_{i+1} for $1 \leq i < s$.

3.1. The Mesh Algorithm

For ease of explanation, we describe the mesh algorithm first, both informally and then as a list of routing rules. The packet-switched version is presented first and is followed by the wormhole version.

3.1.1. Packet Triplex on the mesh

A message can route according to dimension order rules (DO) at any time using restricted buffers (Rules 1 and 2). A message may also take any minimal (Rule 3) or non-minimal route (deroute) (Rule 4) using an empty unrestricted buffer. Moreover, a message that needs to move in the negative direction of the lowest dimension it needs to correct can use any buffer in a dimension greater than the lowest dimension that needs correcting (Rule 5). This includes both minimal and non-minimal routes.

Let l be the lowest dimension that a message still needs to correct. A message selects a path in the mesh at its current location subject to the following routing restrictions.

1. A message may route according to *DO* (using restricted buffers).
2. If a message waits, it waits on any buffer it needs including the one specified by *DO*.
3. A message may use an empty unrestricted buffer on a minimal path.
4. A message may deroute using an empty unrestricted buffer.
5. A message that needs to route in the negative direction of l may use any (restricted or unrestricted) empty buffer in a dimension i, $i > l$. This includes non-minimal routes.

The unrestricted channels provide most of the adaptivity, while the restricted buffers provide deadlock-freedom and in some cases extra adaptivity. The last rule increases the number of routing choices over algorithms like Duato, by allowing messages to violate dimension order routing in the restricted buffers, either with minimal or non-minimal routes. This creates cycles in the buffer dependencies. Thus, there is no acyclic ordering of the restricted buffers as in dimension order routing or Duato. Deadlock is avoided by providing an escape route [Gün81] for every message.

The idea of the deadlock-freedom proof follows. Details can be found in the Appendix. A message always maintains a deadlock-free path in its lowest dimension that needs correcting. A message insures this by following special routing rules for dimension l which are slightly more restrictive than the general routing rules. When a message gets delayed, it avoids deadlock by waiting for and routing in dimension l.

Theorem 1: The packet-switched version of the Triplex routing algorithm, *packet Triplex*, for the mesh is deadlock-free.

3.1.2. Wormhole Triplex on the mesh

The (wormhole) Triplex algorithm is complicated by buffer dependencies caused by arbitrary length messages. Since messages can only wait on restricted buffers, any cycle in the buffer dependencies is created from waiting dependencies between restricted buffers. These dependencies can be direct, resulting from a message in one restricted buffer waiting immediately for another restricted buffer; or they can be indirect, caused by a message which occupies a restricted buffer followed by one or more unrestricted buffers and waits for another restricted buffer. The wormhole algorithm is slightly more restrictive than the packet version. The empty buffer criterion is stricter and Rule 4 is replaced by Rule 4w. Rule 4w states that a message may deroute to an empty unrestricted buffer in a dimension greater than the lowest dimension it needs to correct.

4w. A message may deroute using an empty unrestricted buffer in a dimension i, $i > l$.

Theorem 2: The Triplex routing algorithm for the mesh is deadlock-free.

3.2. The Torus Algorithm

The torus algorithm is more restrictive than the mesh algorithm since the wrap edges pose an additional threat of deadlock. Let l be the lowest dimension in which a message still needs to route. A message m satisfies the *wrap-free* property if m is guaranteed to have a minimal path specified by the routing algorithm to its destination position in dimension l, where the path consists of waiting buffers that never contain a message that uses or has a waiting dependence on the wrap buffers in the negative direction of dimension l. The wrap-free property is easy to determine. For Triplex, the wrap-free property tests whether a message does not use a wrap buffer or has already used a wrap buffer in dimension l in the negative direction. This is sufficient due to the structure of the waiting dependences of the *DO* algorithm. A message selects a path in the torus subject to the same routing restrictions as the mesh, except that Rule 5 is replaced with Rule 5t. Rule 5t permits a message to use any empty buffer in a dimension greater than l provided it needs to correct l in the negative direction, and it does not need or no longer needs a wrap buffer in dimension l. Note, there is nothing special about the negative direction. The algorithm is simply trying to minimize the number of routing restrictions, and this additional flexibility can only be allowed in a single direction. Otherwise, there is a potential for deadlock.

5t. A message that needs to route in the negative direction of l may use any (restricted or unrestricted) empty buffer in a dimension i, $i > l$ if it satisfies the *wrap-free* property. This includes non-minimal routes.

Theorem 3: The packet Triplex routing algorithm for the torus is deadlock-free.

The wormhole Triplex algorithm on the torus is the same as packet Triplex on the torus except that Rule 4 is replaced with Rule 4w.

Theorem 4: The Triplex routing algorithm for the torus is deadlock-free.

For Triplex, any route other than the one specified by *DO* is optional. Thus, the algorithm *minimal Triplex* (*oblivious Triplex*) resulting from restricting Triplex to minimal (oblivious) routes is also deadlock-free. Unless otherwise specified, Triplex refers to the non-minimal wormhole version of the algorithm.

Corollary 5: The minimal Triplex and oblivious Triplex algorithms are deadlock-free for packet and wormhole routing.

Message delivery is in-order since there is a unique path between each source and destination, and with blocking buffering, messages cannot overtake one another.

Corollary 6: Oblivious Triplex delivers messages in-order.

Notice there is no requirement that forces a message to take a non-minimal route. Therefore, livelock-freedom can be achieved in several ways. The first not so interesting alternative is simply to use the minimal adaptive version of the algorithm. The second method allows each message a maximum number of deroutes after which the algorithm reverts to oblivious or some special type of routing, like an Euler path. Counting schemes are used by both [DA93] and [Smi81]. The former counts deroutes as dimension reversals, while the latter accumulates battle scars. Both of these algorithms require a message to carry and update additional routing information in its header, namely the number of misroutes performed. Another related scheme is to assign a time-stamp or fixed priority to each message based on its age [Nga89]. This also requires a message to carry additional routing information in the header. Furthermore time-stamp fields are expensive to compare, since they must be large enough to prevent re-use problems. The fourth alternative is to use a probabilistic scheme like the Chaos router [KS94]. This is simple to implement but results in an algorithm which is probabilistically livelock-free. Nevertheless, this is sufficient in practice. Unlike the Chaos router, the Triplex algorithm is never forced to deroute. Therefore the probability of derouting could be set at machine boot-up, dynamically, or individually based on the specifications of a particular message. In the latter scheme, a message could specify oblivious, minimal, or non-minimal routing at the cost of a few bits in the header. The cost may be reasonable, since routing possibilities can be computed before the selection bits arrive. Individual mode selection might be useful when in-order delivery is required for certain messages and others prefer the flexibility of adaptive routing, or when compile time information is available [Fel93] and a particular type of routing is preferred for the expected traffic.

4. Comparisons

Performance of the Triplex algorithm is explored by simulation. The algorithm is compared to three other routers: the Dally-Seitz oblivious router [DS87], Duato [GPBS94, Dua93], and the Chaos [KS94] router using a flit-level simulator of a 256-node two dimensional torus network. Each of the three routers represents one of the routing classes offered by the Triplex router. The first algorithm provides no adaptivity. The second is a minimal fully adaptive algorithm which allows any shortest path. The Chaos router is a non-minimal fully adaptive router which prefers any minimal path, but occasionally deroutes a message in the presence of severe congestion.

4.1. Router Description

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. The oblivious, Duato, and Triplex algorithms have 2, 3, and 3 virtual channels per channel, respectively yielding a total of 18, 26, and 26 buffers per node for each router, respectively. The oblivious router could be given an extra set of virtual lanes [Dal92], but then it would not support in-order delivery as desired. The Chaos router does not use virtual channels. Instead it uses 10 buffers, in addition to a central queue with 5 buffers, for a total of 15 buffers per node.

Flow control is either wormhole or packet-switched, where the latter uses virtual cut-through routing to avoid the store-and-forward latency penalties typically associated with packet routing. With virtual cut-through routing, a buffer may contain parts of two distinct messages, one that is being received and one that is being transmitted to another buffer.

For wormhole routing, unidirectional (simplex) channels are employed. For packet routing bidirectional (half-duplex) channels shared between the two directions are used. See [Bol93a] for a discussion of the tradeoffs. To maintain a constant channel width between packet and wormhole networks, the unidirectional channels simulated are half as wide as the bidirectional channels, though in the future it may be practical to use each channel in both directions simultaneously [DLD93].

The buffers are one flit long for wormhole routing, while for packet routing, buffers are 20 flits long and buffers can hold an entire message. This is a minimal amount of buffering and greatly increases the throughput achieved by the network. This buffer size also enables us to compare the Triplex algorithm with competitive minimal and non-minimal fully adaptive deadlock avoidance algorithms on the torus[9]. Since a message fits into a buffer entirely, the packet Triplex algorithm is used rather than the Triplex algorithm. Table 1 contains a summary of the differences among the routing algorithms examined.

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 a depth that allows the router cycle time to match that of the channel. The more complex algorithms require

[9]The Chaos router does not support wormhole routing.

Table 1: Summary of the differences between the various routing algorithms.

Router	Cycle Time	Adaptivity	Buffers Required
Oblivious	3	none	18
Oblivious Triplex	4	none	18
Duato	4	minimal adaptive	26
Minimal Triplex	4	minimal adaptive	26
Chaos	4	non-minimal adaptive	15
Triplex	4	non-minimal adaptive	26

a larger pipeline depth resulting in larger node latencies. The node latencies are three cycles for the oblivious and four cycles for the adaptive algorithms [Bol93b]. Although the adaptive algorithms appear quite complex to describe, the additional calculations required beyond that of the oblivious router are actually easy to compute. For example, the Triplex computation determines, in parallel, the dimension and direction of all of the minimal routes, the lowest dimension minimal route. The non-minimal routes are the remaining directions in each dimension. Then, two masks are created, one for the non-minimal routes and the other for all of the minimal routes, except for the lowest dimension minimal route which uses a restricted buffer. When the selection bits arrive, the appropriate mask is combined with the routing choices to obtain oblivious, minimal, or non-minimal routes.

To keep complexity manageable, the router connects at most a single message from an input buffer to an output buffer per cycle. Since all the algorithms are implemented as output-driven routers [FS96], the router computes in parallel the routes needed by each ready message at the head of an input buffer. Then, it selects (at random) one of these messages to move to the empty output buffer being considered.

Before moving a message from an input to an output buffer within a node, the Duato and (wormhole) Triplex algorithms require the full/empty status of the neighboring node's input buffer corresponding to the output buffer under consideration. This status is already present at the node, after a one cycle propagation delay, since it is used for flow control of the channels. Nevertheless, there is a penalty for using this information. The delay in status causes a two flit bubble between consecutive messages for packet and wormhole routing. This reduces the maximum throughput achievable by both the Duato and Triplex algorithms. The other algorithms including the packet Triplex algorithm avoid this penalty, since they only require status about the local output buffers.

4.2. Simulation Parameters

Two lengths of messages are used. For packet routing short messages are 20 flits long. Wormhole routing channels are half as wide as packet channels, and thus use 40-flit short messages for the same amount of data. When a combination of long and short messages are used, the short 40-flit messages are ten times as frequent as the 400-flit long messages.

For non-minimal routers where derouting is forced, excessive derouting is a difficult problem. This occurs in deflection routers which may deroute every cycle; and to a lesser extent, in chaotic routers which must fill a special queue before deroutes are forced. Triplex pre-

vents a message from derouting continuously and wasting bandwidth by routing minimally when possible, requiring a message to wait at least 160 cycles before becoming eligible for deroutes, and by derouting eligible messages with a probability of .1 when no minimal path is available. For packet routing the values used are 400 and .00001, respectively. These parameters were chosen by experimentation which found nearly equivalent results for a large range of parameter values. Probabilistic derouting is also used to guarantee probabilistic livelock-freedom, i.e. as time goes to infinity, the probability a message is still circulating in the network goes to zero. For packet routing the probability of derouting is extremely low. This simply reflects the fact that the non-minimal routing allowed by Triplex is not effective at increasing throughput without degrading throughput or raising latency.

Messages are introduced to each node at every cycle with a probability specified by the applied load (Poisson or constant rate arrivals). The load is normalized by the maximum sustainable load when an average of half the messages cross the network bisection[10], as with uniform random traffic. Thus, for a 256-node two dimensional torus with an average message length of l flits, the maximum possible load is 1 message every $4l$ cycles.

The traffic patterns considered are found in the literature, and are generally thought to be difficult, useful, or both. The following briefly 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 $a_{n-2}a_{n-3}\ldots a_0 a_{n-1}$.
- 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 spots - 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.

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

The simulator is written in C and uses a batch means [Muñ91] method for computing 95% confidence intervals for the expected values of network throughput and latency. The sources of randomness are provided by a prime-modulus, multiplicative congruential generator [LL74] which is considered highly reliable for simulation studies [LO89].

4.3. Results

Simulations were run to examine the performance of Triplex and packet Triplex in each of its modes: oblivious, minimal fully adaptive, and non-minimal fully adaptive. Comparisons were made between Triplex and a single class router using the same class or mode as Triplex. Because Triplex provides all three modes, we do not expect it to perform better than any of the single class routers. Nevertheless, we need to show that the performance sacrificed by using a flexible scheme is not excessive. Results show the Triplex algorithm performs well for the oblivious and minimal adaptive modes, but not as well as expected in the non-minimal mode. Representative results are presented first for the traditional wormhole routing network and then for the packet routing network. The remainder of the graphs can be found in the Appendix.

4.3.1. Wormhole Results

Wormhole results using short messages are presented first. See Figures 1 and 7–8 for the throughput and latency graphs.

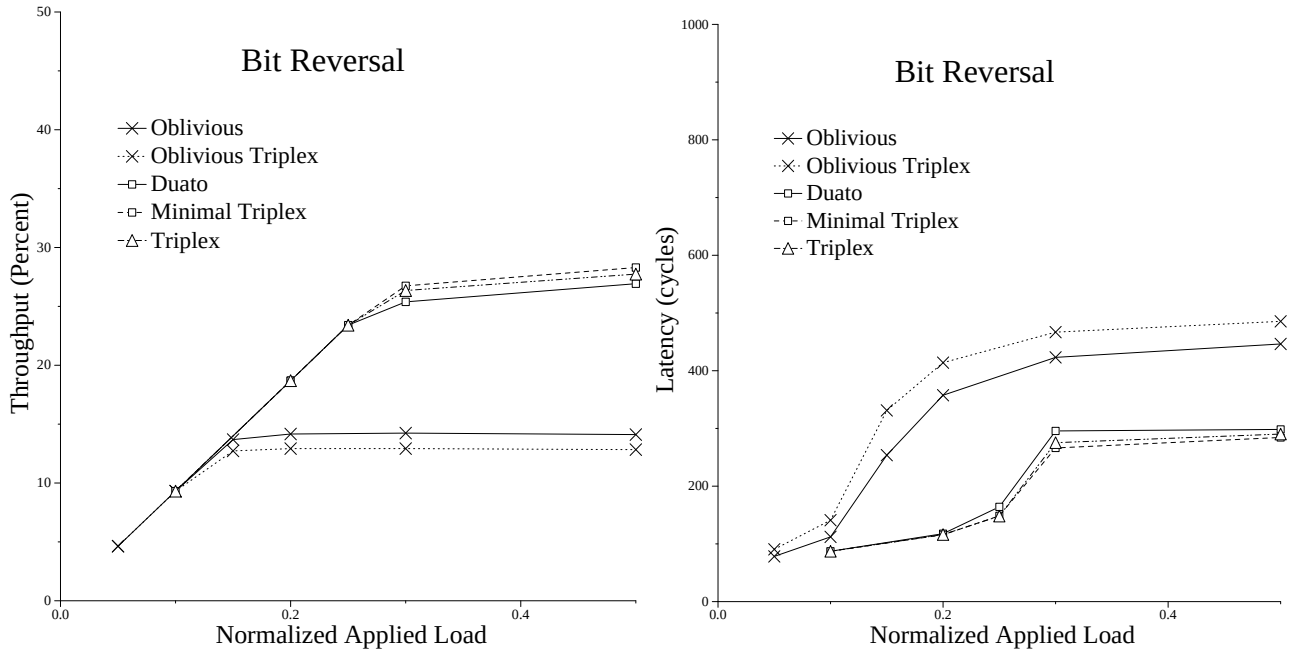

Figure 1: Throughput and latency on 256-node 2D torus with wormhole routing for 40-flit messages.

At very low loads where no congestion is present, all the routers achieve the same throughput. The expected latency depends on the minimal latency of a message through the network.

The oblivious router has the smallest node latency (three versus four cycles); and therefore, has the best expected latency for the different traffic patterns.

At higher loads, the throughput of the oblivious Triplex router does not quite match that of the oblivious router. Oblivious Triplex throughput is within 21 percent of the throughput of the oblivious router at the same applied load. The effect on network throughput is not substantial though, since for almost all of the applied loads, the difference in the normalized throughput achieved is at most 4 percent. The difference in throughput occurs because the oblivious Triplex incurs an extra cycle of latency at each node due to its higher node latency. As expected, oblivious Triplex experiences higher latency than the oblivious router.

The minimal Triplex algorithm exceeded Duato throughput performance in half of the four cases (bit reversal and transpose), while in the other cases (random and hot spot) the Duato achieved a higher throughput, despite its more restrictive routing rules. Latency was slightly lower for the minimal router that achieved the higher throughput. We conjecture that minimal Triplex does not always surpass Duato because Triplex allows the restricted buffers to be used in a manner which violates dimension order routing. This increases the turns a message may make from one dimension to another and hence the number of potential conflicts.

For the non-minimal case, Triplex performs adequately, but not as well as the minimal Triplex algorithm. In the cases were derouting was particularly ineffective (random and hot spot), Triplex also experienced higher latencies than the minimal routers. It is likely that the asymmetries in the virtual channel usage makes the non-minimal routing less effective than it should be. This may be caused by both the non-uniformities from the deadlock prevention scheme and the somewhat restricted non-minimal routes allowed for only a subset of the messages in the network.

When the traffic includes both long and short messages, the results are similar to those with only short messages, though the overall throughput is lower, latency is higher, and the throughput differences between the routers are less distinct. Otherwise, the routers have the same relative performance except for the Triplex algorithm where long messages emphasize the ineffective use of non-minimal routing in the Triplex algorithm. Figures 2 and 9–10 show the throughput and latency of the long and short messages individually. The throughput of the long messages appears to be nearly identical for all of the routers, although a more detailed view would show that the long messages exhibit the same relative differences between routers as the short messages.

Long messages also slightly decrease the *saturation point* of the network, the first normalized applied load (in increments of .05) at which more messages arrive than can be delivered, for each of the traffic patterns and algorithms. See Table 2 for details. The saturation point is important, since after saturation network performance is unpredictable. Although real systems cannot sustain such loads, it provides information about the performance of the system under large bursts of traffic.

4.3.2. Packet Results

This section compares the different routers when packet routing is used. In this case, we can compare packet Triplex to Chaos, a competitive non-minimal packet routing algorithm. See

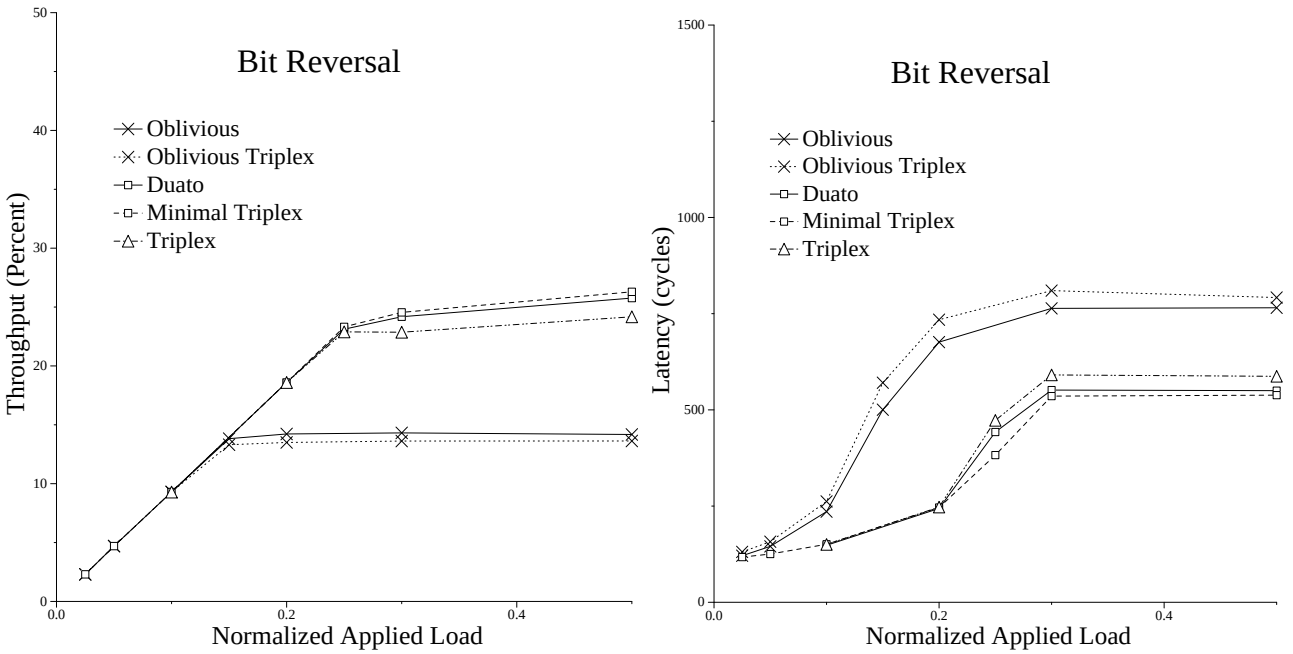

Figure 2: Throughput and latency on 256-node 2D torus with wormhole routing for 400-flit long and 40-flit short messages.

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

Torus Saturation Points for Wormhole Routing										
Traffic	Oblivious		Obliv Triplex		Duato		Min Triplex		Triplex	
Msg type	short	mixed	short	mixed	short	mixed	short	mixed	short	mixed
Random	.20	.20	.20	.15	.30	.25	.30	.20	.30	.20
Bit reversal	.15	.15	.15	.15	.30	.25	.30	.30	.30	.25
Transpose	.20	.20	.20	.20	.25	.25	.30	.25	.30	.25
Hot Spot 1	.20	.15	.20	.15	.25	.20	.25	.20	.20	.20

Figures 3, and 4–6 for the final comparisons between Triplex and the corresponding single class routers.

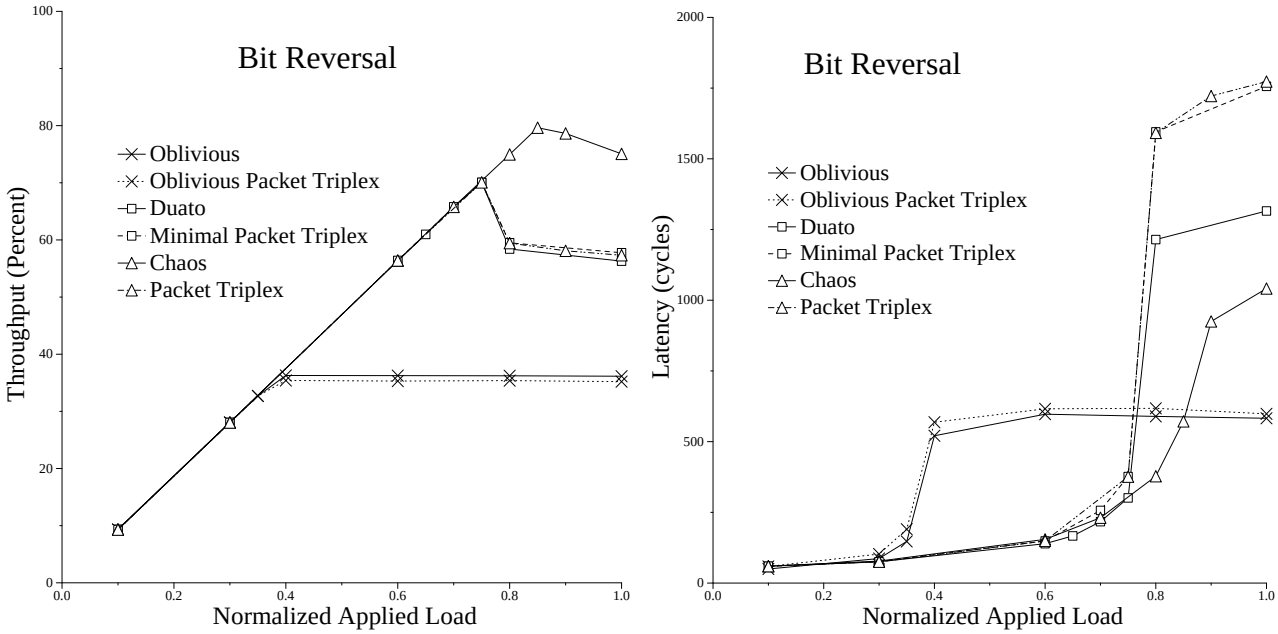

Figure 3: Throughput and latency on 256-node 2D torus with packet routing for 20-flit messages.

As in the wormhole case, at very low loads where no congestion is present, all the routers achieve the same throughput. The oblivious router has the smallest node latency; and thus, has the best expected latency for the different traffic patterns.

At higher loads, the throughput of the oblivious packet Triplex router does not quite match that of the oblivious router. Triplex throughput is within 5 percent (and usually less) of the throughput of the oblivious router at the same applied load while the difference in the normalized throughput achieved is at most 2 percent. As before, the only difference between the two oblivious classes is the extra cycle of latency that packet Triplex incurs at each node. The throughput difference is much greater with the wormhole routed network than the packet network. Due to its higher node latency, the oblivious packet Triplex router experiences a slightly higher latency than the oblivious network.

Although the minimal packet Triplex is less restrictive than the Duato algorithm, it did not match the Duato performance. For all the traffic patterns, Duato matches or exceeds the maximum throughput of minimal Triplex. Nevertheless, for two traffic patterns bit reversal and perfect shuffle, the throughput of minimal Triplex did not degrade nearly as much as Duato. Latency for minimal Triplex is also worse. The only exception was for the perfect shuffle traffic pattern, where the latency of Duato is over twice that of minimal Triplex.

Performance of minimal packet Triplex is improved substantially by omitting (the minimal routes of) Rule 5t. In this case, the two algorithms have the same routing rules but differ in their empty buffer definition. The Duato algorithm has a stricter criterion resulting in bubbles between consecutive messages in the network. Consequently, the throughput of minimal Triplex equals or exceeds (bit reversal and perfect shuffle) that of Duato, although

the additional occupied buffers of Triplex results in higher after saturation latency. We conjecture that Rule 5t, which allows messages to use the restricted buffers according to rules which violate dimension order, permits more turns from one dimension to another in the restricted output buffers. This causes conflicts which impede the flow of messages in a congested network. Maximizing adaptivity seems like a good strategy to increase throughput; however, the results demonstrate that this is not always the case. From here on, we assume that the packet Triplex implementation has omitted the minimal routes of Rule 5t.

For the non-minimal case, packet Triplex performs well, equivalent to the minimal packet Triplex, but has higher latencies than and lacks the sustained throughput achieved by the Chaos router at high loads. We believe that this performance difference arises from the non-uniformities introduced into the network by the routing restrictions that prevent deadlock for the Triplex router. This includes both the underlying oblivious network and the restricted nature of the non-minimal routes. Some messages are not allowed to deroute, while others are but eventually lose this ability. This loss may be beneficial when the message is a hop or two away from its destination [Kon92], but not if it is far from its destination. The Chaos router has none of these non-uniformities, since it relies on the packet-exchange protocol for deadlock prevention [NS89]. The protocol has a simple requirement: if node a sends a message to node b, node a must also accept a message from node b.

Next, packet Triplex was tested with Rule 5t completely omitted. Deroutes were still allowed by Rule 4. For all the cases tested, the throughput of the modified packet Triplex is equivalent to that of the packet Triplex algorithm. Likewise, latency was equivalent or lower than the Triplex algorithm. As before, we believe that allowing some messages to violate dimension order rules, in buffers used primarily for dimension order routing, impedes message flow in congested networks. In the following, we assume that the Triplex implementation has omitted Rule 5t entirely.

Finally, since the Chaos router does not allow messages in an injection buffer to deroute, this feature was disabled in packet Triplex. This reduced the standard deviation of the throughput and latency figures; but otherwise, was insignificant in improving the performance of Triplex.

5. Related Work

A large number of torus routing algorithms have been developed with varying complexity, resource requirements, and switching techniques. Much of this complexity results from resolving the problem of deadlocks in the network. Routing algorithms can be made deadlock-free in one of two ways. The first is by allowing deadlocks to occur and then to recover. Recovery schemes are used in both the compressionless router [KLC94] which uses an abort and retry technique similar to double-buffering [W+88], and in the Disha router [KP95] which has a special set of buffers which a deadlocked message may access exclusively to reach its destination.

The alternative is deadlock avoidance, which is a more commonly used technique to achieve deadlock-freedom. Deadlocks have traditionally been avoided in wormhole routed networks by insuring that the network has a set of connected channels with static acyclic

channel dependencies. Informally, a dependency between two channels occurs when a message can use the one channel followed by the other channel. Dally and Seitz [DS87] presented this idea for deterministic algorithms and Duato [Dua93, Dua95] generalized this idea to adaptive and fault tolerant algorithms. Though most of the deadlock avoidance algorithms share this same underlying principle, they have different resource requirements and varying amounts of adaptivity.

The simplest deadlock-free wormhole algorithm for the torus is the Dally-Seitz oblivious dimension order algorithm [DS87]. This algorithm requires two virtual channels per channel for the torus and restricts a message to correcting the dimensions from lowest to highest. The torus algorithms of Linder and Harden [LH91] and Duato [GPBS94, Dua93] are minimal[11] fully adaptive algorithms, allowing all shortest paths between a message source and destination. The former uses an exponential number of virtual channels per channel, while the latter uses three but requires buffer status from neighboring nodes. Glass and Ni eliminate the use of virtual channels by restricting turns in the network, resulting in a non-minimal partially adaptive algorithm [GN94]. Planar adaptive [CK92] and hierarchical adaptive routing [LC94] are partially adaptive algorithms that sacrifice adaptivity to reduce the crossbar complexity required of fully adaptive algorithms. There are several other algorithms that have been presented, but they do not easily generalize to the torus topology [DA93, DDH$^+$94].

Unlike the previously mentioned approaches, the framework of Schwiebert and Jayasimha [SJ96] does not require a connected network with static acyclic channel dependencies. This results in algorithms with fewer routing restrictions. The Triplex algorithm presented uses this approach.

6. Conclusions

We have presented Triplex, a deadlock-free, non-minimal fully adaptive wormhole routing algorithm for tori. It is the first router that provides the flexibility to change routing classes at system boot-up, dynamically, or individually by message. At the expense of a few extra bits in the header, a message can select oblivious, minimal fully adaptive, or non-minimal fully adaptive routing classes. This may be useful for in-order delivery of certain messages while others prefer the flexibility of adaptive routing, or when compile time information is available and a particular type of routing is preferred for the expected traffic.

Furthermore, Triplex is the first non-minimal fully adaptive wormhole routed deadlock avoidance algorithm for tori. Although the algorithm is non-minimal, no message is forced to take a deroute. The algorithm is also deterministically or probabilistically livelock-free, depending upon the implementation chosen.

Simulations show that the cost of providing a multiple classes does not have a significant performance penalty as compared with the single class oblivious and minimal routers. However, performance of the non-minimal option of Triplex is not much better than the minimal option, making it difficult to justify implementation of the non-minimal class.

[11] The Linder Harden algorithm actually allows a restricted set of non-minimal routes which requires that the minimal or non-minimal decision be made at message injection.

References

[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.

[Bol93a] K. Bolding. Multicomputer interconnection network channel design. Technical Report UW-CSE-93-12-03, University of Washington, Seattle, 1993.

[Bol93b] 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 Arch.*, pages 268–277, 1992.

[CKP+93] D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T. vonEicken. LogP: Towards a realistic model of parallel computation. In *Proc. of Sym. on Prin. and Prac. of Par. Prog.*, May 1993.

[DA93] William J. Dally and Hiromichi Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–75, 1993.

[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.

[DDH+94] W.J. Dally, L.R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. The reliable router: A reliable and highh-performance communications substrate for parallel computers. In *Lecture Notes in Computer Science*, volume 853, pages 241–55, 1994.

[DLD93] L.R. Dennison, W.S. Lee, and W.J. Dally. High performacne bidirectional signalling in VLSI systems. In *Research on Integrated Systems: Proc. of the 1993 Sym.*, 1993.

[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.

[Fel93] E.W. Felten. *Protocol Compilation: High-Performance Communication for Parallel Programs*. PhD thesis, University of Washington, Seattle, WA, 1993.

[FS96] M.L. Fulgham and L. Snyder. A comparison of input and output driven routers. In *Lecture Notes in Computer Science, Proc. of Euro-Par '96*, volume 1123, pages 195–204, 1996.

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

[GPBS94] L. Gravano, G. Pifarré, P.E. Berman, and J.L.C. Sanz. Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1233–1251, 1994.

[Gün81] K.D. Günther. Prevention of deadlocks in packet-switched data transport systems. *IEEE Trans. on Communication*, COM-29:512–524, April 1981.

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

[KLC94] J.H. Kim, Z. Liu, and A.A. Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. In *Proc. of the Intl. Sym. on Computer Arch.*, pages 289–300, 1994.

[Kon92] S. Konstantinidou. Priorities in nonminimal, adaptive routing. In *Intl. Conf. on Par. Processing*, volume I, pages 67–71, 1992.

[KP95] Anjan K.V. and T.M. Pinkston. An efficient, fully adaptive deadlock recovery scheme: DISHA. In *Proc of the Intl. Sym. on Comp. Arch.*, pages 201–210, 1995.

[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. *Simulation Methodology for Statisticians, Operations Analysts, and Engineers*, chapter Uniform Pseudo-Random Variable Generation, pages 65–99. 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.

[Nga89] J. Y. Ngai. *A Framework for Adaptive Routing in Multicomputer Networks.* PhD thesis, California Institute of Technology, Pasadena, CA, May 1989.

[NS89] J. Y. Ngai and C. L. Seitz. A framework for adaptive routing in multicomputer networks. In *Proc. of the Sym. of Parallel Alg. and Arch.*, pages 1–9, 1989.

[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.

[SJ96] L. Schwiebert and D.N. Jayasimha. A universal proof technique for deadlock-free routing in interconnection networks. *Journal of Parallel and Distributed Computing*, 32(1):103–17, 1996.

[Smi81] B.J. Smith. Architecture and applications of the HEP multiprocessor computer sytem. In *Proc. of SPIE*, pages 241–248, 1981.

[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.

[W+88] D. Whobrey et al. A communications chip for multiprocessors. In *Proc. of CONPAR*, 1988.

A. Proofs of Deadlock-freedom

We show the Triplex algorithm is deadlock-free by applying a theorem of Schwiebert and Jayasimha [SJ96]. To keep the paper self-contained, we briefly review in this paragraph the definitions needed for the theorem. A *waiting buffer* is a buffer a message can wait to acquire when all other buffers specified by the routing function cannot be selected. The *buffer waiting graph* (BWG) for a routing algorithm is a directed graph BWG $= (B, E)$ where the vertex set B represents the set of buffers in the network and the edge set E represents pairs of buffers (b_1, b_2) where a message occupying buffer b_1 can wait for buffer b_2. A routing algorithm is *wait-connected* if a message always has at least one waiting buffer.

There is a *direct waiting dependence* between two buffers a and b if a message can use buffer a and wait for waiting buffer b. There is also a *waiting dependence* between a and b if there is a sequence of buffers $(a = b_1, b_2, \ldots, b_s = b)$ such that there is a message m_i that uses buffer b_i and waits for waiting buffer b_{i+1} for $1 \leq i < s$. This is equivalent to having a path from a to b in the buffer waiting graph being considered.

Theorem 1: [SJ96] If a routing algorithm, R, is wait-connected and the BWG for R is acyclic, then R is deadlock-free.

Next we define the terminology and notation needed. Let DO be the dimension order Dally-Seitz oblivious, deadlock-free wormhole routing algorithm [DS87]. This algorithm is minimal, makes routing decisions independent of the message source and input channel, and is suffix closed[12]. Furthermore, this algorithm is wait-connected, deadlock-free, and has an acyclic BWG. Let DO_i be the restriction of Dally Seitz algorithm to dimension i. The restricted algorithm DO_i also makes routing decisions independent of dimensions other than i.

The DO algorithm routes a message from the lowest dimension 0 to the highest dimension $n-1$ by applying DO_0, then DO_1, ..., and finally DO_{n-1}, where the direction in each dimension is chosen to make the message route minimal. Although the specific DO_i rules differ slightly for the mesh and the torus[13], the particular details are not relevant. Hence, the mesh algorithm will not be distinguished from the torus algorithm except by context.

Let b_i^+ (b_i^-) denote the buffers corresponding to a virtual channel in the positive (negative) direction of dimension i. The direction will only be specified when a distinction between the positive and negative direction is necessary. The distinction between an input buffer b_i^{in} in dimension i and an output buffer b_i^{out} in dimension i will only be made for packet routing, and only when necessary. When using packet routing, the terms packet and message are used interchangeably.

For convenience, we assume that for all the routing algorithms presented, if a message waits, it waits on the buffer specified by DO. For packet routing, a packet in an input buffer waits on the appropriate buffer in the lowest dimension it needs to correct, while, a packet in an output buffer must wait on the corresponding input buffer (there is no other choice). We proceed by showing that each algorithm has an acyclic BWG, and hence by Theorem 1 is deadlock-free. Later, we show how to remove this waiting restriction.

A.1. The Mesh Algorithm

For ease of explanation, we describe the mesh algorithm first. The packet-switched version is presented first and is followed by the wormhole version.

A.1.1. Packet Triplex on the mesh

In packet routing each message in a cycle occupies a single buffer, regardless of whether store-and-forward or virtual cut-through flow control is used. Thus waiting dependencies between buffers occur only when a message in one buffer waits directly for another restricted buffer in the same or neighboring node. Furthermore, since a packet in an input buffer can only wait on a restricted buffer, a cycle in the BWG only contains restricted buffers.

[12]Informally, every path a message takes to a particular destination through a node a can be used by a message injected at node a to reach the same destination.

[13]The torus algorithm uses two virtual channels per direction per dimension, while the mesh uses one.

Fact 1: For packet (wormhole) routing, when a message waits in an input () buffer, it waits for the restricted buffer specified by *DO* which is in the *minimal* direction of l, the lowest dimension it needs to correct.

Lemma 2: Given a cycle in the BWG where l is the lowest dimension (input and output) buffer in the cycle, the lowest dimension any message needed to correct when it was routed to the output buffer corresponding to the input buffer it holds in the cycle is l.

Proof: Consider a cycle in the BWG where the lowest dimension in the cycle is l. A message in a cycle only occupies one buffer. So if a message m in a cycle needed to correct a dimension lower than l, when it was routed to the output buffer corresponding to the input buffer it holds in the cycle, m would wait on this dimension. Thus, l would not be the lowest dimension in the cycle. □

Lemma 3: Given a cycle in the BWG where l is the lowest dimension (input and output) buffer in the cycle, all input buffers in the cycle in dimension l have been used minimally according to *DO*.

Proof: Consider a cycle in the BWG where l is the lowest dimension buffer in the cycle. Let m be a message in a restricted (all buffers in the cycle are restricted) input buffer b_l^{in} in dimension l in the cycle. By Lemma 2 dimension l was the lowest dimension m needed to correct when it was routed to restricted output buffer b_l^{out} corresponding to input buffer b_l^{in}. And by definition of the routing algorithm, message m was routed to its restricted buffer in dimension l by *DO* rules, which are minimal. □

Lemma 4: There are no cycles in the BWG where the lowest dimension in the cycle is used in the positive and negative direction.

Proof: Assume there is a cycle in the BWG. Let l be the lowest dimension of the cycle, and assume l is used in the positive and negative direction. Then, there exists a message m in an input buffer in the cycle that is not in the positive direction of dimension l, which waits for a restricted output buffer b_l^+ in the positive direction of l. We show that message m violates the routing rules, and hence no such cycle exists. There are two cases. The first case assumes m resides in a restricted buffer in a dimension greater than l, and the second assumes that m occupies a restricted buffer in the negative direction of dimension l. With packet routing, there are no other cases to consider, since each message in a cycle resides in exactly one restricted buffer.

First, message m holds a restricted input buffer b_h in the cycle, for some dimension $h > l$, and waits for restricted output buffer b_l^+ in the positive direction of dimension l. By Fact 1, this direction is minimal. By Lemma 2, l is the lowest dimension m needed to correct when m was routed to buffer b_h. But m is not allowed to use a restricted buffer b_h in a dimension greater than l, when it needs to route in the positive direction of l.

Second, message m occupies a restricted input buffer b_l^- in the negative direction of dimension l in the cycle, and waits for the restricted output buffer b_l^+ in the positive direction of dimension l. By Lemma 3, m was routed minimally in the negative direction to its current input buffer b_l^- in the cycle. And by Fact 1, m waits in the minimal direction for b_l^+ which is positive. This is a contradiction. □

Theorem 5: The packet-switched version of the Triplex routing algorithm, *packet Triplex*, for the mesh is deadlock-free.

Proof: The algorithm is wait-connected since a message in a buffer can always wait on the waiting buffer specified by DO, which is wait-connected.

A cycle must use both directions of each dimension on a mesh. Thus, it follows from Lemmas 2, 3, and 4 that the BWG is acyclic. And by Theorem 1, the algorithm is deadlock-free, since the BWG is wait-connected and acyclic. □

A.1.2. Wormhole Triplex on the mesh

The (wormhole) Triplex algorithm is complicated by buffer dependencies caused by arbitrary length messages. Since messages can only wait on restricted buffers, any cycle in the buffer dependencies is created from waiting dependencies between restricted buffers. These dependencies can be direct, resulting from a message in one restricted buffer waiting immediately for another restricted buffer; or they can be indirect, caused by a message which occupies a restricted buffer followed by one or more unrestricted buffers and waits for another restricted buffer. Thus, a message in a cycle occupies at least one restricted buffer and waits for another restricted buffer. Furthermore, the waiting dependencies between two restricted buffers do not have to be in the same or a neighboring node.

Another complication of the longer messages, is that the lowest dimension in a cycle in the BWG is no longer guaranteed to be the lowest dimension in the original cycle in the network. Thus, we need to refer to the network cycle to reason about the routing decisions made.

Lemma 6: Given a cycle in the network corresponding to a cycle in the BWG where l is the lowest dimension buffer in the network cycle, the lowest possible dimension any message needed to correct when it was routed to a buffer it holds in the cycle is l.

Proof: Let m be a message in a network cycle corresponding to a cycle in the BWG. Also let l' be the lowest dimension message m needed to correct when it was routed to a buffer b it holds in the cycle. Furthermore, suppose l' is less than the lowest dimension l in the cycle. There are two possibilities. Either m corrected dimension l' sometime after acquiring buffer b, or m still needs to correct dimension l' and waits on a buffer in dimension l' in the cycle (a message must wait on the lowest dimension it needs to correct). Both, however, contradict the assumption that l is the lowest dimension buffer in the cycle. (Nevertheless, a message blocked in the cycle may have a tail that extends beyond the cycle and holds a buffer in a dimension lower than l.) □

Lemma 7: Consider a cycle in the network corresponding to a cycle in the BWG. Let l be the lowest dimension in the network cycle. Then all buffers in the network cycle in dimension l have been used minimally according to DO or by using unrestricted buffers in dimension l.

Proof: Consider a cycle in the network corresponding to a cycle in the BWG. Let l be the lowest dimension in the network cycle. By Lemma 6, l is the lowest possible dimension

any message needed to correct when it occupied any buffer it currently holds in the cycle. A message can only deroute in a dimension greater than the lowest dimension that it needs to correct, which for messages in the cycle is at least as great as l. Thus, a message in the cycle was routed minimally when obtaining its buffers in the cycle in dimension l; and by definition of the routing algorithm, the buffers were chosen according to DO rules or by using unrestricted buffers in dimension l. □

Definition: A positive *turn* buffer is a buffer b_l^+ in the positive direction of dimension l in a cycle which resides in a node which has the smallest[14] position or offset in dimension l of any node in the cycle.

Lemma 8: There are no cycles in the BWG where l, the lowest dimension in the corresponding network cycle is used in both the positive and negative direction.

Proof: Assume there is a cycle in the BWG. Let l be the lowest dimension in the corresponding network cycle, and suppose l is used in both the positive and negative directions. Then, there exists a message m which holds a buffer in the cycle that is not in the positive direction of l and later waits for or uses a positive turn buffer b_l^+ in the cycle, We show that message m violates the routing rules, and hence no such cycle exists. There are three cases.

First message m holds a restricted b_h buffer in the cycle, in some dimension h, $h > l$ and later waits for or uses and holds positive turn buffer b_l^+ in the positive direction of dimension l. By Fact 1, this direction of l is minimal for m. By Lemma 6, l is the lowest dimension m needed to correct when m used buffer b_h. But m is not allowed to use a restricted b_h buffer in a dimension greater than l, when it needs to route in the positive direction of l.

Second, message m holds a (restricted or unrestricted) b_l^- buffer in the cycle in the negative direction of dimension l and later waits for or uses and holds positive turn buffer b_l^+ in the cycle. By Lemma 7, message m was routed minimally to all buffers it holds in the cycle in dimension l. If m holds b_l^+ and b_l^-, this is a contradiction since m cannot be routed minimally by being routed first in the negative, and then in the positive direction. If m holds b_l^- and waits for b_l^+, Fact 1 says that m waits in the minimal direction; and again, this is a contradiction.

Third, message m does not satisfy either of the above cases. There are two possibilities. If m waits for positive turn buffer b_l^+, all the buffers m holds are unrestricted buffers in dimensions greater than l, and m is not part of the cycle. If m holds positive turn buffer b_l^+, all the buffers m holds after positive turn buffer b_l^+ are unrestricted buffers in dimensions greater than l. Hence, there must be another message in the cycle that waits on positive turn buffer b_l^+. □

Theorem 9: The Triplex routing algorithm for the mesh is deadlock-free.

Proof: The algorithm is wait-connected since a message can always wait on the waiting buffer specified by DO, which is wait-connected.

A cycle must use both directions of each dimension when routing on a mesh. Thus, it follows from Lemmas 6, 7, 8, and Theorem 1 that the BWG is acyclic, and the algorithm is deadlock-free. □

[14] Without loss of generality, we assume the nodes are ordered in row major order.

A.2. The Torus Algorithm

The proof for the torus is similar to the mesh but requires three additional lemmas to show that cycles are not created by routing on the wrap edges. Since a message that needs to correct the lowest dimension l in the negative direction has more freedom than one that needs the positive direction of l, we need to consider two separate cases, one where dimension l is used only in the negative direction, and the other where l is used in the positive direction. We start with the base case, which considers a cycle in a single dimension l and follow with the more general case. Again, the packet algorithm is presented first.

A.2.1. Packet Triplex on the torus

Lemma 10: There are no single dimension cycles in the BWG.

Proof: Assume there is a cycle in the BWG that only uses buffers in dimension l. By Lemma 3, all messages in input buffers in the cycle in dimension l were routed minimally by DO rules using restricted buffers. DO is suffix-closed, and independent of the current input buffer. So given a message m in an input buffer in the cycle, there exists some packet that always follows DO routing which could reside in m's buffer and wait for the same buffer as message m in the cycle. The output buffers in the cycle can be filled identically. Thus in this case, the edges or waiting dependencies in the BWG are identical to those of DO which is acyclic. □

Lemma 11: On the torus, there are no multi-dimensional cycles in the BWG which use the lowest dimension in the cycle in the positive direction only.

Proof: Assume there is a multi-dimensional cycle in the BWG which uses l, the lowest dimension in the cycle in the positive direction only. Then, there exists a message m that holds a restricted input buffer b_h in the cycle, in some dimension h, $h > l$ and waits for an output buffer b_l^+ in the positive direction of dimension l. By Fact 1, this direction in dimension l is minimal for m. By Lemma 2, l is the lowest dimension m needed to correct when m used buffer b_h. But m is not allowed to use restricted buffer b_h in a dimension greater than l, when it needs to route in the positive direction of l. □

Lemma 12: On the torus, there are no multi-dimensional cycles in the BWG where the lowest dimension in the cycle is used in the negative direction only.

Proof: Assume there is a multi-dimensional cycle in the BWG where l, the lowest dimension in the cycle, is used in the negative direction only. Since dimension l is used in a single direction, the cycle must use a wrap buffer w in dimension l. Also, there exists a restricted input buffer b_h in some dimension h, $h > l$, in the cycle immediately before the wrap buffer w, but excluding buffers in dimension l. Let m be the message that holds input buffer b_h. Since a message in a cycle occupies a single restricted buffer, m waits on a restricted output buffer in dimension l at or before the wrap buffer w. But m is not allowed to use a restricted buffer in a dimension greater than l unless m is guaranteed to have a minimal path to its destination position in dimension l, where the path consists of waiting buffers which never have waiting dependencies on the wrap buffers in dimension l (wrap-free property). □

Theorem 13: The packet Triplex routing algorithm for the torus is deadlock-free.

Proof: The algorithm is wait-connected since a message can always wait on the waiting buffer specified by DO, which is wait-connected. The result follows from Lemmas 2, 3, 4, 10, 11, 12, and Theorem 1. □

A.2.2. Wormhole Triplex on the torus

Lemma 14: There are no single dimension cycles in the BWG in dimension l, the lowest dimension of the corresponding cycle in the network.

Proof: Assume there is a single dimension cycle in the BWG which only uses restricted buffers in dimension l, the lowest dimension in the corresponding cycle in the network. If buffers in dimensions greater than l are used in the cycle, they must be unrestricted buffers.

By Lemma 7, all the buffers in the cycle in dimension l were routed minimally by DO rules or by using unrestricted buffers in dimension l. The waiting dependencies of DO are acyclic. Routing by DO_l is independent of dimensions other than l. Thus taking unrestricted buffers in dimensions other than l does not cause a cycle in dimension l in the BWG. These routes only add the following edges. If there exists a route from node a to node b in dimension l, using an unrestricted buffer in a dimension i other than l results in an edge in the BWG from node a to b', where b' is a node with the same positions as b, except in dimension i.

Furthermore, since DO is suffix-closed and independent of the current buffer (and whether its restricted or unrestricted), routing a message minimally in dimension l either by DO or unrestricted buffers does not alter the subsequent route or waiting buffer used by the message in dimension l, as specified to DO rules. So routing in unrestricted buffers in dimension l only adds edges in the BWG from an unrestricted buffer in dimension l to restricted waiting buffers specified by DO. There are no edges from restricted buffers to unrestricted buffers. Thus, there can be no cycle in dimension l. □

Lemma 15: On the torus, there are no cycles in the BWG which contain a dimension greater than l, the lowest dimension in the corresponding cycle in the network, provided that l is used in the cycle in the positive direction only.

Proof: Assume there is a cycle in the BWG containing a restricted buffer in some dimension h, $h > l$, where l is the lowest dimension in the corresponding cycle in the network. Also suppose l is used in the network cycle in the positive direction only. Then, there exists a message m that holds a restricted buffer b_h in the cycle, and later waits for or uses and holds a buffer b_l^+ in the positive direction of dimension l. By Fact 1, this direction in dimension l is minimal for m. By Lemma 6, l is the lowest dimension m needed to correct when m used buffer b_h. But m is not allowed to use a restricted b_h buffer in a dimension greater than l, when it needs to route in the positive direction of l. □

Lemma 16: On the torus, there are no cycles in the BWG which contain a dimension greater than l, the lowest dimension in the corresponding cycle, provided that l is used in the negative direction only.

Proof: Assume there is a cycle in the BWG containing a restricted buffer in some dimension h, $h > l$, where l is the lowest dimension in the corresponding cycle in the network. Also suppose l is used in the negative direction only. Since dimension l is used in a single direction, the cycle must use a wrap buffer w in dimension l. Let b_h be a restricted buffer in dimension h, in the cycle immediately before the wrap buffer w, but excluding unrestricted buffers and buffers in dimension l. Let m be the message that holds buffer b_h. Message m cannot wait for an unrestricted buffer, so one of the following describes m. Message m waits on a restricted buffer in dimension l at or before the wrap buffer w. Alternatively, m uses the wrap buffer w in dimension l and waits on a restricted buffer in the cycle after the wrap edge. Both are impossible. First, m is not allowed to use a restricted buffer in a dimension greater than l unless m is guaranteed to have a minimal path to its destination position in dimension l, where the path consists of waiting buffers which never have waiting dependencies on the wrap buffers in dimension l (wrap-free property). Second as a consequence of the wrap-free property, m cannot use restricted buffers in a dimension greater than l, if it needs to use a wrap buffer in dimension l. □

Theorem 17: The Triplex routing algorithm for the torus is deadlock-free.

Proof: The algorithm is wait-connected since a message can always wait on the waiting buffer specified by DO, which is wait-connected. The result follows from Lemmas 6, 7, 8, 14, 15, 16, and Theorem 1. □

It is not necessary to restrict a message to wait on the buffer specified by DO. A message may actually wait on all the buffers it needs. To see this, we define a subgraph BWG' of the BWG for each algorithm as follows. Let BWG' be the subgraph of the BWG obtained by removing all the edges from the restricted buffers to the unrestricted buffers. Furthermore, remove all edges between restricted buffers that violate DO routing. The resulting buffer waiting graph BWG' is still wait-connected. The proof is similar to those presented, except that the following definition and theorem are needed instead. A *True Cycle* is a cycle in the BWG which can be created without the simultaneous use of any buffer.

Theorem 18: [SJ96] A routing algorithm, R, that allows a blocked message to wait for multiple output buffers is deadlock-free iff R is wait-connected for some subgraph BWG' of BWG and BWG' has no True Cycles.

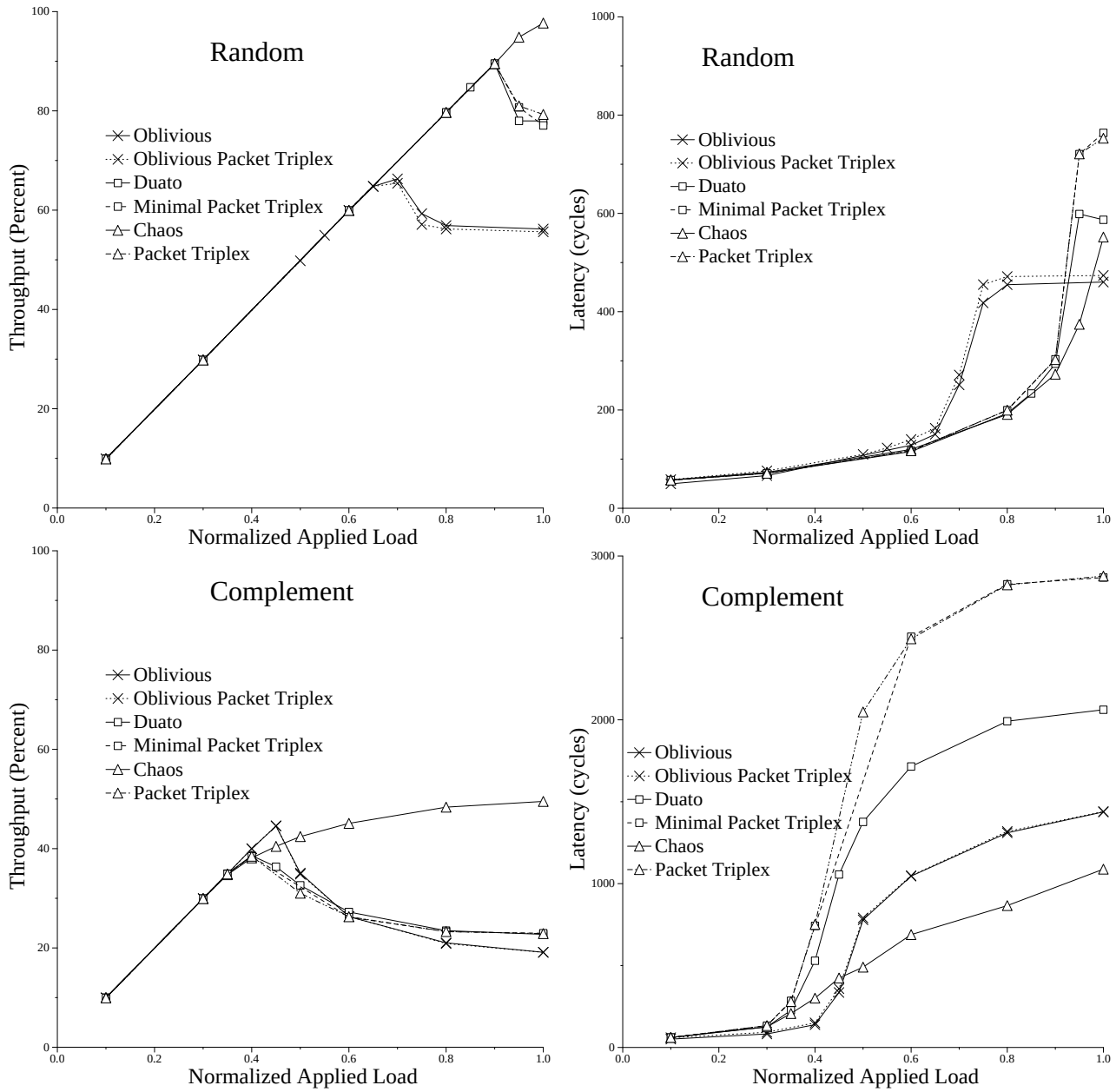

Figure 4: Throughput and latency on 256-node 2D torus with packet routing for 20-flit messages.

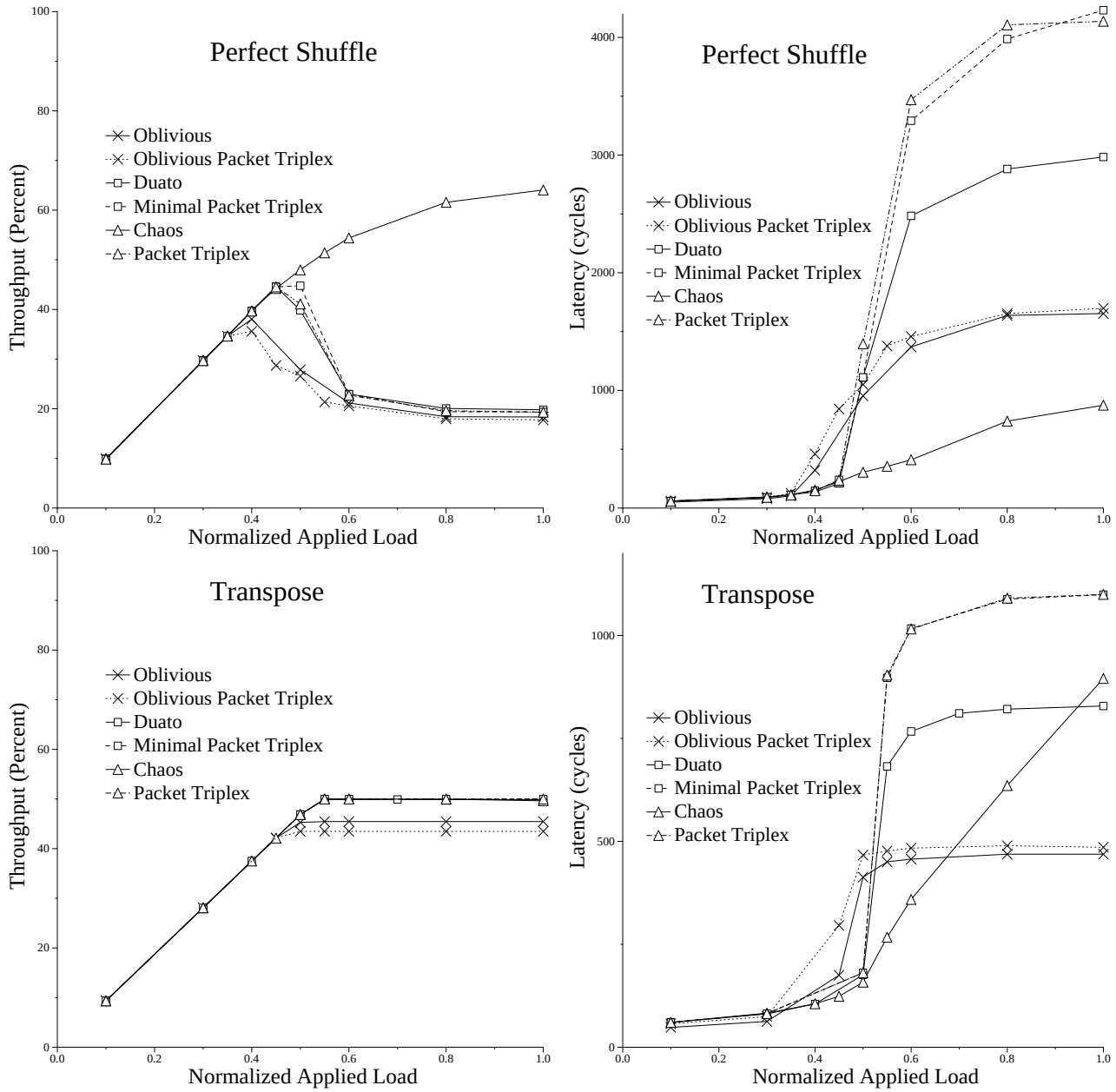

Figure 5: Throughput and latency on a 256-node 2D torus with packet routing for 20-flit messages.

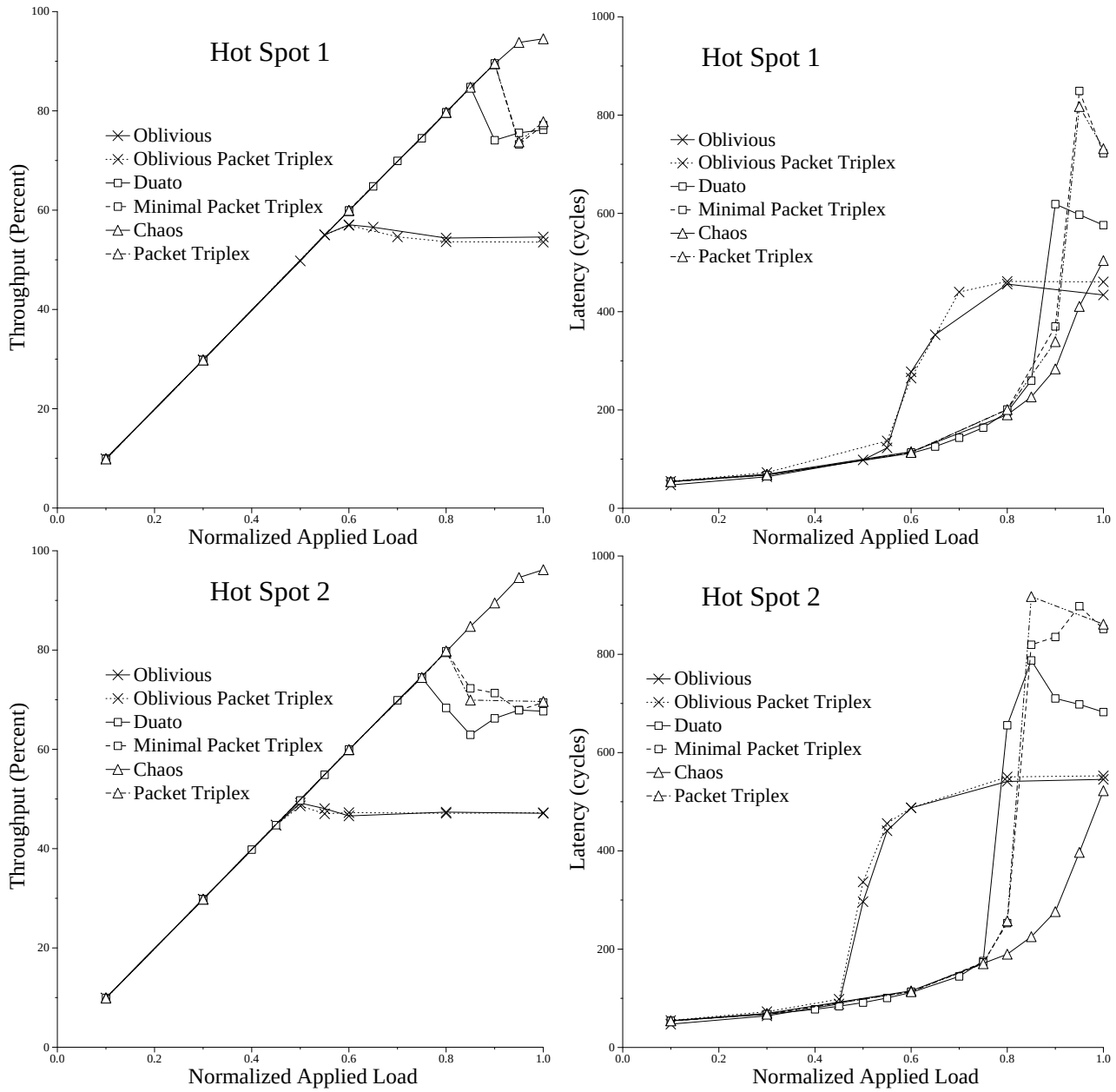

Figure 6: Throughput and latency on a 256-node 2D torus with packet routing for 20-flit messages.

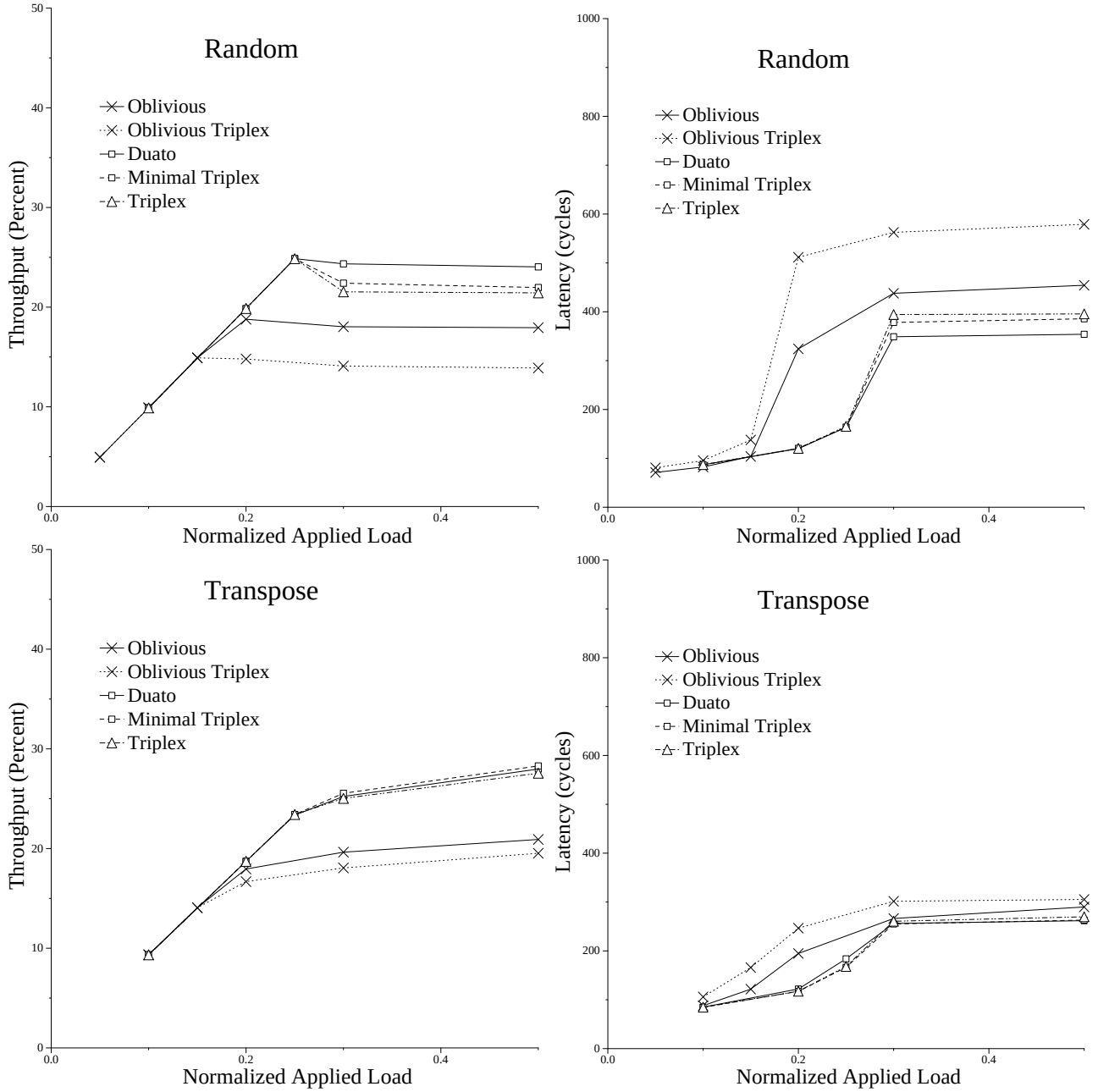

Figure 7: Throughput and latency on a 256-node 2D torus with wormhole routing for 40-flit messages.

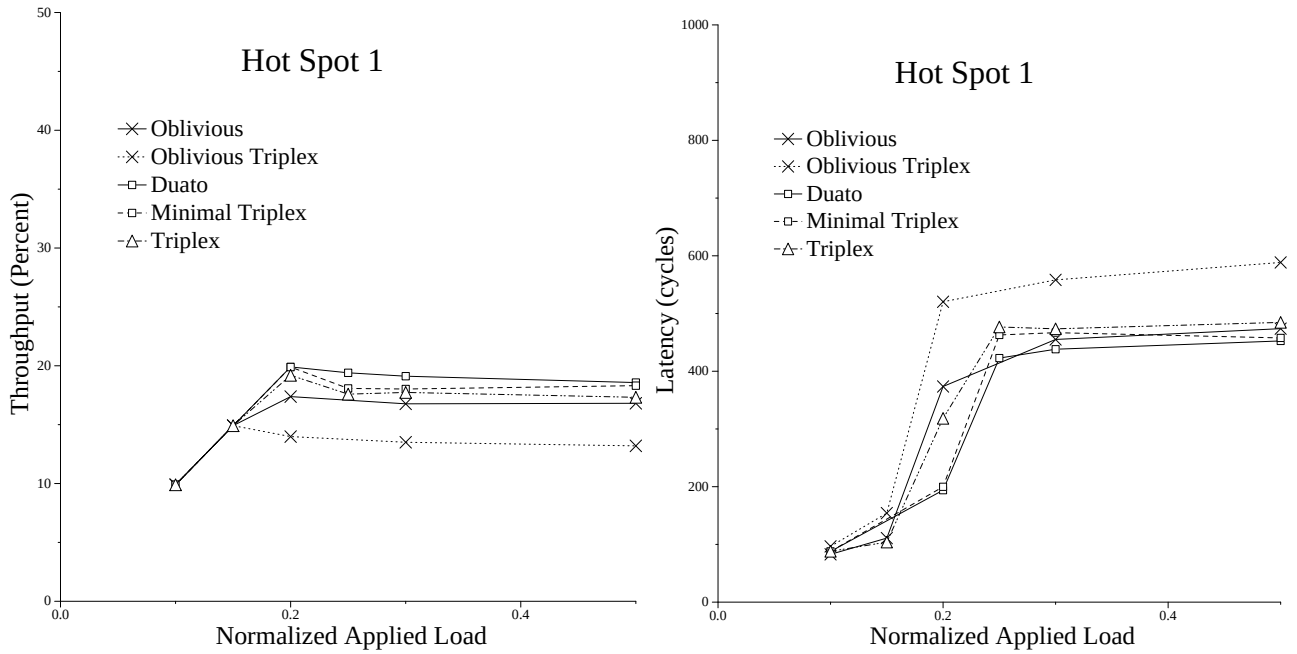

Figure 8: Throughput and latency on a 256-node 2D torus with wormhole routing for 40-flit messages.

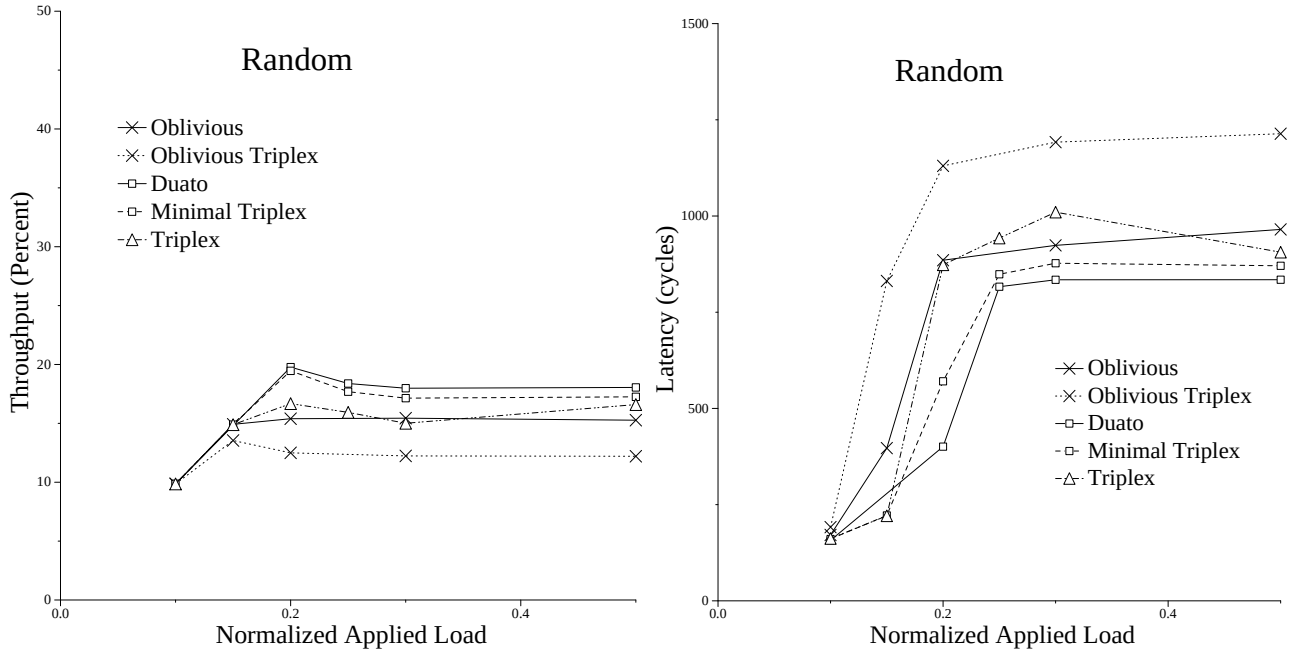

Figure 9: Throughput and latency on a 256-node 2D torus with wormhole routing for 40-flit short and 400-flit long messages.

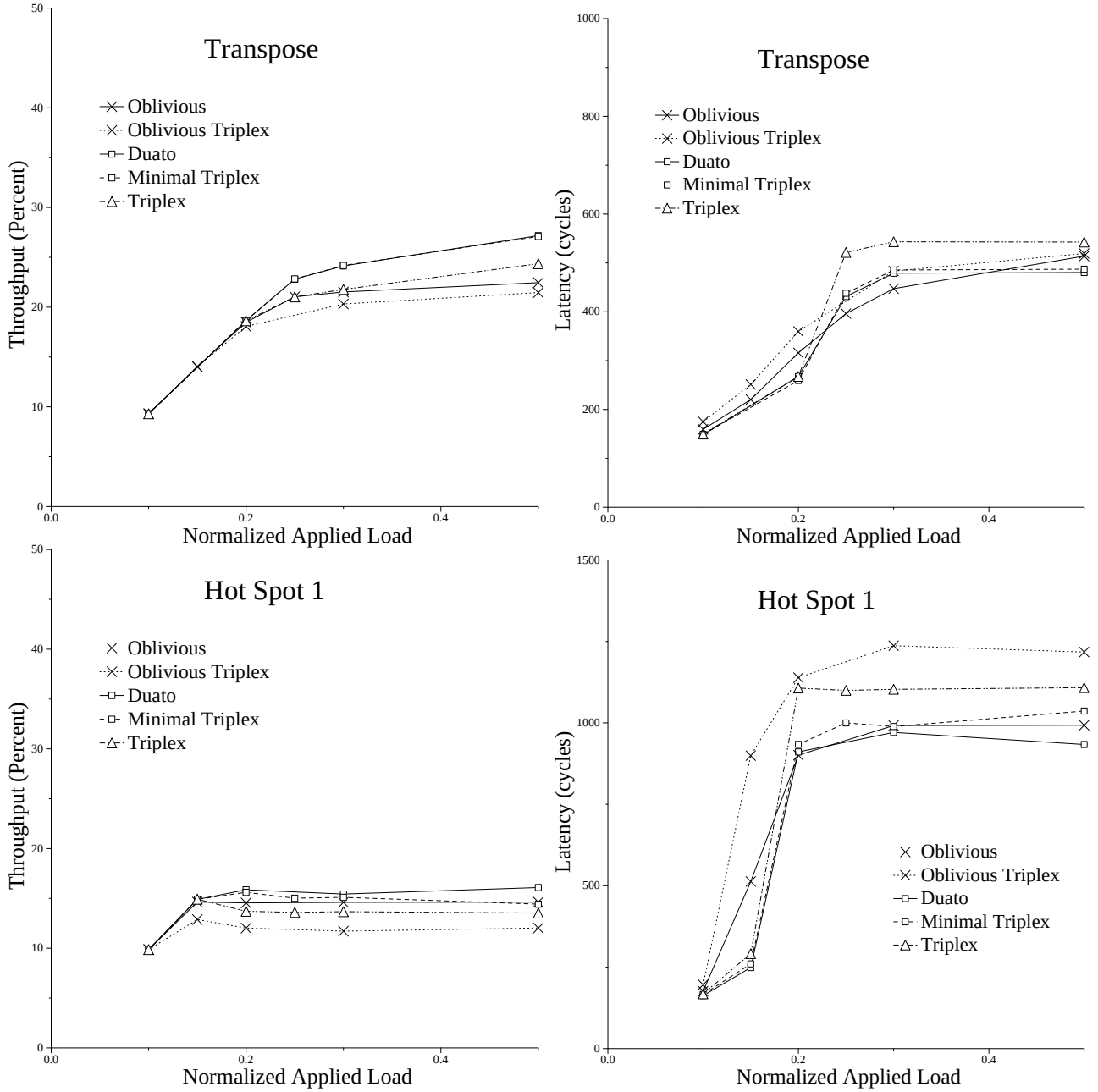

Figure 10: Throughput and latency on a 256-node 2D torus with wormhole routing for 40-flit short and 400-flit long messages.