Performance of Chaos and Oblivious Routers Under Non-uniform Traffic

Melanie L. Fulgham [*]
mel@cs.washington.edu
Lawrence Snyder [†]
snyder@cs.washington.edu
University of Washington

July 23, 1993

Abstract

Chaos router is a nonminimal adaptive packet router that is both deadlock free and probabilistically livelock free [Kon91]. Unlike its predecessors [Nga89], the Chaos router uses randomization to provide an efficient way to avoid livelock [KS90]. The Chaos router has been compared in software simulation studies to the state-of-the-art oblivious routers (i.e. dimension order routers) on the hypercube topology [KS91], and the mesh and torus topologies [BS92]. The results, based on simulations of two different workloads, one uniform and one non-uniform load, show that in general the Chaos router is superior to the oblivious router. However, two workloads alone are not representative of the typical types of traffic encountered in practice. In this paper five additional non-uniform workloads are simulated. Each was chosen to represent one of the various types of communication encountered in typical programs. The workloads, compared on both the 256 node torus and hypercube topologies, show considerable variety and reveal insight on the effectiveness of Chaotic adaptive routing on non-uniform loads. Data presented includes the saturation point of the networks under the various loads, the throughput as a function of presented load, latency, and total delay of packets through the network. Collectively, this data gives a much clearer and complete characterization of the Chaos and oblivious routers' behavior on non-uniform traffic.

Keywords: hypercube, torus, chaos, adaptive routing, simulation, deroute.

[*]This work is supported by an NSF Graduate Fellowship and ONR grant N00014-91-J-1007.
[†]This work is supported in part by NSF grants MIP-9013274 and MIP-9213469.

1 Introduction

The Chaos router is a deadlock free, probabilistically livelock free, nonminimal adaptive packet router for dynamically injected network traffic [Kon91]. It was introduced as a practical alternative to oblivious and priority routers. Unlike its predecessors [Nga89], the Chaos router uses randomization to provide an efficient way to avoid livelock [KS90]. Using a simplified functional design, the Chaos router has been compared in software simulation studies to the state-of-the-art oblivious routers (i.e. dimension order routers) on the hypercube topology [KS91], and the mesh and torus topologies [BS92]. The results demonstrate a general superiority of the Chaos router on the two workloads simulated. However considered alone, the two workloads are not representative of the typical traffic patterns encountered in practice.

One of the workloads considered in the original studies is the *uniform random* traffic pattern. Despite its wide use in packet routing studies, it is often criticized (though there are exceptions [Per92]) as not being representative of workloads encountered in practice. Typical programs exhibit locality and although little is understood about the characteristics of practical workloads, there is no reason to believe accesses occur randomly unless the target machine hashes addresses randomly. However, even the effectiveness of hashing is not understood. The second workload considered previously, called the *4X hot spots* traffic pattern, is a non-uniform pattern in which ten randomly selected nodes are four times more likely to be the destination of packets than the other nodes of the network. Such a load creates delivery hot spots [Kon91] and is intended to represent situations in which program variables, such as synchronization variables, are frequently referenced.

In this paper five new non-uniform workloads are considered. In addition, the 4X hot spots and the uniform random workload are repeated in order to gather additional data. The workloads, simulated on 256 node torus and hypercube topologies, display considerable variety and reveal insight into the effectiveness of the Chaos router on non-uniform loads. Data is presented on the saturation point of the networks under the various loads, the throughput as a function of presented load, latency, and total delay of packets through the network.

Besides the above mentioned results, this paper is the first to present saturation profiles of any router. The saturation data demonstrates several unexpected features.

- The oblivious hypercube router saturates at very small loads for all non-uniform traffic patterns except for the complement. The complement has anomalous behavior and is discussed in more detail later.
- The difference between the normalized saturation load of the oblivious and the Chaos routers can be enormous; e.g. for the transpose on the hypercube .10 versus .70 respectively.
- The Chaos torus saturates with random traffic just before the theoretical limiting load. The router does nearly as well for the more difficult bit reversal permutation.
- The Chaos router saturates before the oblivious router in only two cases: the complement on the torus and a few of the hot spot traffic patterns on the hypercube.

Although saturation is usually not reported, it is very important since once a router is saturated, the delay through the network becomes infinitely large and hence unpredictable. The saturation data presented, when combined with the standard throughput and latency measures, produces a much clearer characterization of routing behavior.

The results in this paper are from simulation rather than from analysis or direct measurement of hardware. Analysis is precluded by the current intractability of modeling dynamic loads with non-uniform traffic patterns. Hardware results will not be available for several years. Simulation is often (justifiably) maligned because practitioners produce numbers purported to represent reality, though on closer inspection the model is oversimplified. Furthermore, the reported numbers can be of dubious mathematical reliability. The simulations performed for this paper are designed to overcome these criticisms. The simulations mimic the behavior of the UW chip designs at the level of the system clock for both of the routers. One time unit in the simulation corresponds to one clock period in the hardware design. In addition, a theoretically sound simulation methodology has been followed [Muñ91]. Confidence intervals are provided for all the primary data. We believe that no more accurate data can be produced, short of building four 256 processor parallel computers, one for each of the two routers and two topologies.

The paper is organized as follows. Section 2 reviews the oblivious and Chaos routers for the hypercube and torus topologies. The simulation methodology is described in Section 3. Results of the comparisons between the two routers are presented in Section 4 and, concluding remarks are in Section 5.

2 Router Review

This section briefly describes the various parameters used in this study. The two routers considered are the oblivious router and the Chaos router, both of which have been studied extensively in the cited literature.

The routers have $d + 1$ bidirectional channels, where $d = 4$ for the 256 node torus and $d = 8$ for the 256 node hypercube. Although the channels are bidirectional, they are capable of transmitting in only one direction at a time. The direction may change only at the end of a packet. Each channel is connected to a pair of buffers each capable of holding a single packet. For the d channels connected to other nodes in the network, the two buffers are called the *input frame* and the *output frame*. The remaining channel connects the router to its processor by the *injection frame* and the *delivery frame*. The Chaos router has, in addition to the pairs of frames, a small buffer called the *multiqueue* which is composed of $d + 1$ packet frames.

The routers are virtual cut-through [KK79]. Therefore once the header of a packet arrives at a node, it can be routed immediately (cut-through) to a free channel before the packet has completely arrived. Packets can also cut-through the Chaos multiqueue frames.

The packets are composed of 20 flits, or FLow control unITs. Both routers can send or receive a flit in 1 time unit on each channel. The Chaos router can transmit a flit to and/or from each multiqueue frame in unit time. For the oblivious router 2 time units are charged to process the header flit. The Chaos router processes the header in 3 time units. This implies that the node latency, the minimum time to transit a node and one channel, is 3 cycles for the oblivious and 4 cycles for the Chaos router[1]. These times were calculated assuming that the destination address fits in the first flit. This implies, for these experiments, that a flit must transmit at least 8 bits. If multiple packet headers arrive at a node simultaneously, they are routed one at time. This simplification is justified [Kon91] because the added complexity of processing packets in parallel slows the logic. In addition, the frequency with which it is beneficial to process multiple packets is small, given 20 flit packets and dimension $d \leq 8$.

The oblivious router sends packets towards their destinations by "correcting the dimensions" in order, lowest to highest. If the channel for the lowest dimension needed is unavailable, the packet must wait. For the torus, packets route to the column matching the x dimension first, and then route to the row specified by the y dimension. Virtual channels are used to prevent deadlock [DS87]. For the hypercube, where the node addresses are represented by binary numbers, the dimensions of the current location are matched with the dimensions of the destination address from the least significant bit to the most significant bit. No additional deadlock prevention is needed for dimension order routing on the hypercube.

Since the Chaos router is adaptive, it is not restricted to dimension order routing. Channels can either be productive, taking a packet closer to its destination, or nonproductive, taking a packet farther from its destination. When a packet is sent farther from its destination, it is called a *deroute*.

The Chaos router processes packets according to the current channel dimension which changes in a round-robin fashion to the next interesting dimension. A dimension i is *interesting* if output frame i is empty and a packet in either the multiqueue or one of the input frames needs dimension i. If the current output frame is interesting, the router does one of the following in order of precedence:

- Selects the oldest packet in the multiqueue that needs the current dimension output channel. If such a packet exists, does the following. Moves it to the current output frame, and if the current dimension input frame is not empty, reads this packet into the multiqueue.
- Randomly selects an input dimension containing a packet that needs the current output channel. If the selected input dimension is the same as the current output dimension, moves the selected packet to the current output frame. If the selected input dimension is not the current dimension, and the current dimension input is not empty, do the following. If the multiqueue is full, deroute a random packet in the multiqueue by moving it to the current output frame. Read the current dimension input packet into the multiqueue. If the selected input dimension is not the current dimension and the current dimension input is empty, move the selected input packet to the current dimension output frame.

The use of randomness is critical to the operation of the Chaos router [KS90]. First, randomness is used to select a packet in the multiqueue for derouting. This is essential, since it assures that the router is probabilistically livelock free. Livelock occurs in nonminimal adaptive routers when a packet continually circulates in the network. The second use of randomness is optional but is thought to be effective for distributing the load. Randomness is used to select one of the set of input frames including the injection frame that contain packets that need the current output dimension.

[1]Previous comparisons between oblivious and Chaos routers [Kon91, KS90] used a 1:1 ratio rather than a 4:3 ratio, as well as less precise modeling of internal details, which explains the minor discrepancies in the reported results.

Mechanisms incorporated into nonminimal adaptive routers to assure deterministic livelock freedom typically introduce complexity, significantly slowing the router operation. However by using randomness the Chaos router incurs only a small routing time penalty with respect to the simple and fast oblivious router. Though probabilistic livelock freedom is a weaker condition than the deterministic livelock freedom, it's believed the two are operationally equivalent.

3 Methodology

The results were derived using the Chaos project simulator. The program is written in C and uses a simulation cycle time corresponding to one time unit which is the time to transmit a flit across a channel. The sources of randomness are provided by the Learmonth-Lewis prime-modulus, multiplicative congruential generator [LL74] which is considered highly reliable for simulation studies [LO89].

It is assumed that the system satisfies the Function Central Limit Theorem. This allows the use of the batch means method [Muñ91] for computing 95% confidence intervals. See Glynn and Iglehart [GI90] for the theoretical justification.

All nodes generate packets with destinations specified by the simulated traffic pattern. The rate of packet generation is called the presented load. All loads in this study are normalized to the maximum theoretical load for uniform random traffic, defined to be the load at which the network bisection[2] becomes saturated. For the 256 node hypercube and torus with 20 flit messages, the maximum injection rate is one message every 20 and 80 cycles respectively.

The following quantities are measured:

- *Throughput*, the rate the network delivers packets.
- *Source queuing time*, the time a packet waits from creation until it is injected into the injection frame.
- *Latency*, the time a packet spends in the network, from the time the first flit enters the injection frame until the last flit enters the delivery frame at the destination.
- *Delay*, source queuing + latency.
- *Saturation*, the smallest load at which more packets are created than delivered.
- *Hops*, number of edges (channels) a packet traverses to reach its destination.
- *Shortest path distance*, the minimum number of hops required to deliver a packet.
- *Total deroutes*, (hops - shortest path distance)/2. Every deroute moves a packet one hop farther from its destination and one hop is needed to correct the deroute.

[2]A network bisection is the smallest set of channels crossing a hyperplane separating the network into equal halves.

Though this terminology is largely consistent with the literature, variations do occur.

The traffic patterns considered have been used previously in the literature and are generally thought to be difficult, useful or both. Following is a description of the traffic patterns simulated. Let the binary representation of the source node be $a_{n-1}a_{n-2}\ldots a_0$. Also, let $\overline{0} = 1$ and $\overline{1} = 0$.

- *Random*, all destinations including the source are equally likely.
- *4X Hot Spots*, ten randomly selected nodes are distinguished. Destinations are chosen randomly such that the distinguished nodes are four times more likely to be chosen than the undistinguished nodes.
- *Complement*, is a permutation where each source node sends packets to $\overline{a_{n-1}a_{n-2}\ldots a_0}$.
- *Transpose*, is a permutation where each node sends packets to $a_{n/2-1}a_{n/2-2}\ldots a_0 a_{n-1}a_{n-2}\ldots a_{n/2}$.
- *Bit Reversal*, is a permutation where each node sends packets to $a_0 a_1 \ldots a_{n-1}$.
- *Shuffle*, is a permutation where each node sends packets to $a_{n-1}a_{n/2-1}a_{n-2}a_{n/2-2}\ldots a_{n/2}a_0$.
- *Random Leveled*, each node with $i < n/2$ bits set to one sends a packet to a randomly selected node $b_{n-1}b_{n-2}\ldots b_0$ with i one bits satisfying $b_{n-1}b_{n-2}\ldots b_0 \mid a_{n-1}a_{n-2}\ldots a_0 = 0$ where $\mid$ is the bitwise AND operator. Nodes with $i \geq n/2$ one bits simply choose a random destination with i one bits.

All nodes generate packets at the rate specified by the presented load. When routing a permutation, a source node always generates packets with the same destination. The other traffic patterns generate a destination for each packet when it is created.

The traffic patterns illustrate different features. As mentioned earlier the random traffic is simply a standard benchmark used in network routing studies. The 4X 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 since it requires all packets to cross the network diameter in the hypercube and the network bisection in both topologies. Given x and y axes through the center of a torus network, the complement destination is the composition of the x and y axes reflection of the source. Transpose and bit reversal are important because they occur in practical computations and can cause worst case behavior in hypercubic oblivious routers [Lei92]. The shuffle converts row major indexing to shuffled row major indexing on the mesh or torus topologies. This indexing scheme can be used for efficient sorting [TK77]. Random leveled traffic can cause worst case behavior for oblivious hypercubes [VB81].

4 Experimental Results

4.1 Saturation

The first set of results simply identifies the saturation points for the different traffic patterns. The saturation point reported is the first applied load, using intervals of .05, that saturates the network.

Saturation				
	Hypercube		Torus	
Traffic	oblivious	Chaos	oblivious	Chaos
Random	0.60	0.70	0.65	0.95
Transpose	0.10	0.70	0.55	0.55
Bit reversal	0.15	0.70	0.40	0.85
Shuffle	0.35	0.75	0.55	0.70
Random level	0.20	0.70	0.50	0.55
Complement	0.50	0.55	0.45	0.35

Figure 1: Minimum load at which saturation is detected.

See Figure 1. Saturation does not necessarily occur at the knee of the throughput, latency, or delay curves since the effects of saturation can alter system behavior before during, or after saturation depending on the particular conditions.

After saturation some system statistics such as network delay are no longer valid since they do not have a limiting distribution. The load where the system saturates is an important measure since after saturation there is no way to predict the delivery time of messages.

The traffic patterns exhibit considerable diversity in the throughput they are able to sustain. In all cases on the hypercube except several of the hot spot traffic patterns which will be discussed later in more detail, the Chaos router saturates at a higher load than the oblivious router. The Chaos torus also saturates at an equal or higher load than the oblivious torus for all the traffic patterns except for the complement permutation. In both cases the difference is modest. The differences between the two routers can be great, as illustrated by the hypercube transpose permutation.

The differences can also be indistinguishable as in the torus transpose. This implies that, for this traffic pattern, no benefit is gained from relaxing the requirement of dimension order routing. The transpose on the torus is a reflection of the source about the line $y = -x$ given a coordinate system through the center of the network. This pattern causes a continuous hot spot along the diagonal of the network for both the Chaos and oblivious torus routers.

The behavior of the complement permutation is especially interesting. On the oblivious hypercube the complement achieves an unusually high throughput when compared to the other non-uniform traffic patterns. To understand why this happens note that for the oblivious router, dimensions are traversed from lowest to highest only. This implies that input frame i has packets destined for output frame $i+1$. At loads close to saturation most of the input frames should be occupied since the complement packets traverse all dimensions and hence use all channels equally. Therefore when the oblivious router is selecting a packet to use output channel i, it almost always finds a packet; specifically, the packet in input frame $i-1$. More importantly, no other packet in the node has a conflict with the selected packet because only packets in input frame i need output frame $i+1$. The complement on the Chaos torus is more complex. Each packet is destined for a node that is the composition of the x axis and y axis reflection of the source in a coordinate system passing through the center of the network. This causes a hot spot in the center of the network and about the wrap around links at the ends of each of the axes, (if viewed in 3-D, this would be like having two centrally located hot spots at opposite sides of the network), preventing the Chaos torus from reaching the maximum possible throughput. However, for the complement the oblivious torus excels over the Chaos because it has fewer conflicts. Each packet suffers conflicts only with packets being injected in the current direction of travel and at the one node where the packet turns from correcting the x dimension to the y dimension.

When comparing the complement with the other non-uniform traffic patterns, for both the hypercube and torus, the major limiting factor of the complement permutation is the bisection bandwidth. All the packets in the complement must cross the network bisection; whereas, the same is not true of the other traffic patterns. In addition, on the hypercube the complement must also travel the network diameter.

4.2 Throughput and Latency

In this section the behavior of the two routers is compared by examining expected throughput and expected latency for the seven traffic patterns. The graphs display the presented load versus either throughput or latency. Ninety-five percent confidence intervals for these statistics are shown by connecting the left endpoints and by connecting the right endpoints of the confidence intervals for each load. Confidence intervals are not visible for measurements with very small error. The first method we use to compare the two routers is to examine their throughputs.

Throughput for the Chaos router is greater than or equal to the oblivious for both topologies for all traffic patterns at all loads except for a few of the hot spot traffic patterns on the hypercube. However the difference is small, less than a few percent of the normalized throughput. See Figures 4 and 5 in the Appendix. A more detailed discussion of hot spot traffic follows in Section 4.3.

The throughputs for the transpose on the Chaos hypercube and the bit reversal permutation on the Chaos torus are especially noteworthy. The transpose on the Chaos hypercube saturates with a normalized throughput of 68%, whereas the oblivious saturates at a throughput of 9%. The throughput of the oblivious router at the load where the Chaos saturates is only 16%. With the bit reversal permutation, the throughput of the torus Chaos router at saturation is more than double the throughput of the oblivious router from about 39% to 82% respectively.

After saturation, the throughput of the Chaos hypercube degrades for the bit reversal, transpose, shuffle, and random leveled traffic patterns. However, the throughput still remains higher than the oblivious router. Additional data not shown here, shows that optimal queue size depends upon the traffic pattern and the particular goals of the router. In general a multiqueue of size $d + 1$ performs well for all the traffic patterns simulated. Larger multiqueues are able to sustain greater throughputs, but increase latency and latency variance after saturation. Latency is unaffected before saturation due to the cut-through feature of the multiqueue.

Throughput on the oblivious hypercube router does not degrade significantly. This is the primary strength of the oblivious hypercube router. In a saturated oblivious system, messages are forced to wait for needed channels. We believe the derouting capability of the Chaos router is no longer beneficial when the network is saturated and very congested. In this case, waiting for the desired channels is more effective than using bandwidth to deroute messages. The oblivious torus router does not share this strength. On the torus the oblivious throughput degrades after saturation with the complement, random, shuffle, and random leveled traffic patterns. This is most likely due to the asymmetry in the oblivious network introduced by the virtual channels used for deadlock protection in the torus [Bol92, AV92].

Derouting also affects latency. In general the torus latencies have three phases. When the load is below the neighborhood of the saturating load of the oblivious torus, latency for the Chaos torus router is slightly greater for all the traffic patterns. This is due to the higher latency charged to cross the Chaos node. After the oblivious torus saturates, the Chaos torus experiences lower latency until it saturates. After saturation the Chaos torus router experiences much greater latency than the oblivious router due to both the higher throughput of the Chaos and the increased path lengths of the messages caused by derouting. The one exception is the complement where it is still more beneficial to deroute than to wait.

Latency on the Chaos hypercube has two phases. Before the Chaos hypercube saturates, the Chaos hypercube has a lower or comparable latency than the oblivious hypercube. The greater adaptivity in the hypercube paths allows the Chaos hypercube to hide the larger latency needed to cross the Chaos node. After saturation the Chaos hypercube router experiences much greater latency for the same reasons as the Chaos torus. See Figures 6 and 7 in the Appendix.

Before either of the routers saturate, the delays are comparable to the latencies. Therefore for the Chaos torus, delays are slightly higher than the oblivious torus. On the hypercube, the Chaos has equivalent or slightly lower delays than the oblivious router. After either router reaches saturation, delay comparisons are meaningless.

4.3 Hot Spots

For hot spot traffic, placement of the hot spots affects performance, particularly on the oblivious routers which cannot route around the hot spots. Simulations were run for six torus and eight hypercube hot spots arrangements. See the Appendix for the location of the hot spots and for the arrangement of the hot spots on the torus. The hot spot patterns were chosen to demonstrate the variety of behavior in hot spot traffic and are not necessarily representative of a "typical" set of 8 hot spot patterns which have been generated randomly as specified by the hot spot traffic definition.

We are primarily interested in the case when all the hot spots are on distinct nodes. In this case, the oblivious torus does the best when the hot spots are evenly distributed resulting in a peak of about 60% of the normalized throughput (case 4). However when the hot spots are clustered throughput degrades slightly (cases 2, 5, 6) and when arranged in a linear fashion, the oblivious torus throughput degrades significantly, peaking only at 47% (case 3). The lack of adaptivity, especially for packets that must traverse a row or column of hot spots, is particularly detrimental to oblivious throughput. The Chaos torus does much better than the oblivious torus reaching between 85% and 90% of the normalized throughput for all the hot spot cases simulated. See Figure 10 in the Appendix.

When two of the hot spots are on the same node, the difference between the two routers is not so dramatic (case 1). The Chaos torus does about 5% better than the oblivious torus and reaches 55% of the normalized throughput. In this case, throughput is affected by the delivery capacity of the double hot spot node.

For the hypercube, the arrangement of the hot spots is more difficult to visualize. We consider two experiments, the first with the standard delivery rate and the second with the delivery rate of each node increased by the hot spot factor. With the standard delivery rate, the hot spot nodes become a bottleneck since they cannot accept packets destined for them fast enough. In the Chaos, this causes packets waiting to be delivered to a hot spot to deroute and consume network bandwidth. The standard delivery rate results in slightly worse throughput than the oblivious hypercube.

With the faster (quadrupled) delivery rate, the Chaos hypercube peaks at a throughput of about 50% before degrading slightly below 40% for all the hot spot cases except case 2, which peaks at about 42%. The oblivious does almost as well for the randomly placed hot spots reaching throughput between 36% and 38%. However when the hot spots form a tight cluster (case 7) or are connected in two contiguous paths (case 8), the oblivious router cannot even reach 30% of the normalized throughput. See Figures 8 and 9 in the Appendix.

When there is more than one hot spot at a hypercube node, the two routers have lower throughput with similar performance. However the Chaos hypercube does slightly better than the oblivious with the standard delivery and the oblivious hypercube does better with the quadrupled delivery. We expect that the Chaos would peak at a higher throughput than the oblivious hypercube if the delivery rate of the double hot spot were increased to match the expected arrival rate of the packets.

Latencies follow the same general phases as for the other traffic patterns and will not be discussed here. See Figures 11, 12, and 13 in the Appendix for more details.

Saturation figures are as expected. The more difficult the hot spot arrangement is for the router the lower the saturation point. The Chaos router saturates later for all the hot spots cases except three (case 1, 2, and 3) with the standard delivery rates. This inferior performance is a result of delivery bottlenecks at the hot spot nodes, as explained above. When the delivery rate of each node is quadrupled (4X d.r.), the Chaos saturates at a higher load than the oblivious in all the hot spot cases.

4.4 Discussion

For supersaturating loads, the Chaos hypercube benefits from a little extra help. We believe the key to getting good performance, predictable delivery times, and throughput that doesn't degrade is to prevent the router from overloading the network after saturation has occurred. Overloading results in many deroutes which increase message latency and reduce the bandwidth available. Improvement can be achieved in two ways. The easiest, but not the most practical, is to increase the queue size. This allows the network to handle a larger applied load before too many deroutes occur. A more practical solution is to throttle the injection of messages so that excess derouting does not occur. Input throttling can be approximated several ways such as by monitoring local congestion and by using a global controller to combine the data and inform the nodes of a needed change in the local injection policies. The Chaos torus does not suffer from throughput degradation problems.

Saturation				
	Hypercube		Torus	
Traffic	oblivious	Chaos	oblivious	Chaos
1	0.20	0.15	0.55	0.55
1 4X d.r.	0.25	0.35	0.60	0.95
1 8X d.r.	0.25	0.40		
2	0.25	0.20	0.50	0.90
2 4X d.r.	0.35	0.50		
3	0.25	0.20	0.50	0.90
3 4X d.r.	0.35	0.55		
4	0.25	0.25	0.65	0.90
4 4X d.r.	0.40	0.55		
5	0.25	0.25	0.55	0.90
5 4X d.r.	0.40	0.60		
6	0.25	0.25	0.55	0.90
6 4X d.r.	0.35	0.55		
7	0.25	0.25		
7 4X d.r.	0.30	0.55		
8	0.25	0.25		
8 4X d.r.	0.25	0.55		

Figure 2: Minimum load at which saturation is detected for various hot spot traffic.

5 Conclusions and Future Work

We have compared the Chaos router to an oblivious router on a 256 node hypercube and torus using various traffic patterns. We do not necessarily expect all of these traffic patterns to be used in practice. However, we believe they are representative of the types of traffic a router encounters. Routing a variety of traffic patterns also demonstrates the wide range of behaviors that can be exhibited by a particular router.

This study shows that the Chaos router performs better than the oblivious router provided that the Chaos hypercube router is not used for continual supersaturating loads. This is reasonable since most network traffic is thought to be bursty and running long periods with a saturated network makes message delivery unpredictable in any model. The Chaos router saturates at the same or higher applied load for all the traffic patterns and topologies except for the complement on the torus and a few of the hot spot traffic patterns on the hypercube using the standard delivery rate. In all the cases the the difference is modest. When the delivery rate of each node is increased to match the arrival rate of the packets, the Chaos hypercube saturates after the oblivious for all the hot spot traffic patterns. The Chaos has greater or equal throughput than the oblivious router for all of the traffic patterns on both topologies. The only exception is for hot spot traffic with the standard delivery rate on the hypercube where the difference is less than a few percent of the normalized throughput. Again, if the delivery rate is increased as above, the Chaos hypercube throughput is superior to that of the oblivious router.

Latency on the Chaos hypercube router is comparable to or better than the latency of the oblivious router for all cases before saturation approaches on the Chaos. The torus does not have as much available adaptivity, resulting in lower latency for the oblivious torus before the oblivious router saturates. After the oblivious torus saturates and before the Chaos saturates, the Chaos torus has lower latency. After Chaos saturates, the Chaos has higher latency than the oblivious for all but one of the traffic patterns on the two topologies. This is due to both the higher throughput the Chaos maintains and to the cost of the deroutes.

Delays on the routers are comparable to the latencies. Therefore, the Chaos torus has slightly higher delays than the oblivious router while the Chaos hypercube has comparable or slightly lower delays than the oblivious hypercube. Note that delay comparisons are valid only before either of the routers has saturated.

For supersaturating loads, the Chaos hypercube benefits from a little extra help. To prevent network overloading which results in throughput degradation, we recommend either increasing the queue size or throttling the input.

Besides higher throughput, the Chaos router has the advantage that it is fault tolerant. Unlike oblivious routers, adaptive routers do not necessarily break in the presence of faults. Work has already been started to insure that no message can get trapped in a faulty Chaos network [BS91].

Acknowledgements

We thank Smaragda Konstantinidou and Kevin Bolding for writing initial versions of the simulator, Gerry Shedler for help with the simulation methodology, and Carl Ebeling for many invaluable discussions.

References

[AV92] Vikram S. Adve and Mary K. Vernon. Performance analysis of multiprocessor mesh interconnection networks with wormhole routing. Technical Report 1001a, Univ. of Wisconsin – Madison, June 1992.

[Bol92] Kevin Bolding. Non-uniformities introduced by virtual channel deadlock prevention. Technical Report 92–07–07, University of Washington, Seattle, WA, July 1992.

[BS91] K. Bolding and L. Snyder. Overview of fault handling for the chaos router. In *Proc. of the 1991 IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, November 1991.

[BS92] K. Bolding and L. Snyder. Mesh and torus chaotic routing. In *Proc. of The Advanced Research in VLSI and Parallel Systems Conference*, 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.

[GI90] Peter W. Glynn and Donald L. Iglehart. Simulation output analysis using standardized time series. *Mathematics Operations Research*, pages 1–16, February 1990.

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

[Kon91] Smaragda Konstantinidou. *Deterministic and Chaotic Adaptive Routing in Multicomputers*. PhD thesis, University of Washington, Seattle, WA, May 1991.

[KS90] S. Konstantinidou and L. Snyder. The chaos router: A practical application of randomization in network routing. In *Proc. of the 2nd ACM Symp. on Parallel Algorithms and Architectures*, pages 21–30, 1990.

[KS91] S. Konstantinidou and L. Snyder. Chaos router: Architecture and performance. In *Proceedings of the 18th International Symposium on Computer Architecture*, pages 212–221. IEEE, May 1991.

[Lei92] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1992.

[LL74] G.P. Learmonth and P.A.W. Lewis. Statistical tests of some widely used and recently proposed uniform random 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.

[Muñ91] David Muñoz. Multivariate standardized time series in the analysis of simulation output. Technical Report TR-68, 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.

[Per92] Michael J. Pertel. A critique of adaptive routing. Technical Report CS-TR-92-06, California Institute of Technology, Pasadena, CA, 1992.

[TK77] C.D. Thompson and H.T. Kung. Sorting on a mesh-connected parallel computer. *Communications of the ACM*, 20(4):263–271, 1977.

[VB81] L. G. Valiant and G.J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th ACM Symposium on Theory of Computing*, pages 263–277, 1981.

A Hot Spot Locations

Following are the hot spot nodes for each of the hot spot cases.

1. 146 102 94 51 196 25 107 94 15 224

2. 61 12 8 245 5 27 69 28 98 46
3. 3 239 207 83 6 9 89 125 7 255
4. 77 241 105 197 98 126 223 251 163 52
5. 223 251 163 52 74 220 70 179 55 158
6. 210 225 243 73 149 241 136 227 130 88
7. 0 1 2 4 8 16 32 64 128 3
8. 0 1 3 7 15 129 131 135 143 128

Figure 3 shows the arrangment of the hot spot traffic patterns for the torus with cases 1-6 arranged left to right, top to bottom.

B Simulation Results

Figures 4 - 13 containing the throughput and latency results for both the oblivious and Chaos routers on the 256 node hypercube and the 256 node torus. Notice that the scale of the axes may differ from graph to graph in order to show the data more clearly. The dashed and dotted lines represent the upper and lower bounds of the 95% confidence intervals for the data. If no such lines are visible, the interval is very small. In the graphs, the traffic patterns noted in the legends are abbreviated as follows: *r.* is random traffic, *b.p.* is the bit reversal permutation, *c.p.* is the complement permutation, *t.p.* is the transpose permutation, *s.f.* is the shuffle permutation, *h.* is hot spot traffic, and *r.l.* is random leveled traffic. For the hot spot traffic the legend *4X DR* means the delivery rate is quadrupled. The graphs are labeled in the Figure captions from left to right, top to bottom.

C Simulation Data

Figures 14 and greater contain tables with actual simulation data used to plot the graphs. The midpoint and half length of the 95% confidence intervals for throughput and latency are reported. An asterisks following data means the percent error of the 95% confidence interval exceeds three percent. The abbreviation *xput* stands for throughput and *h. l.* for half length.

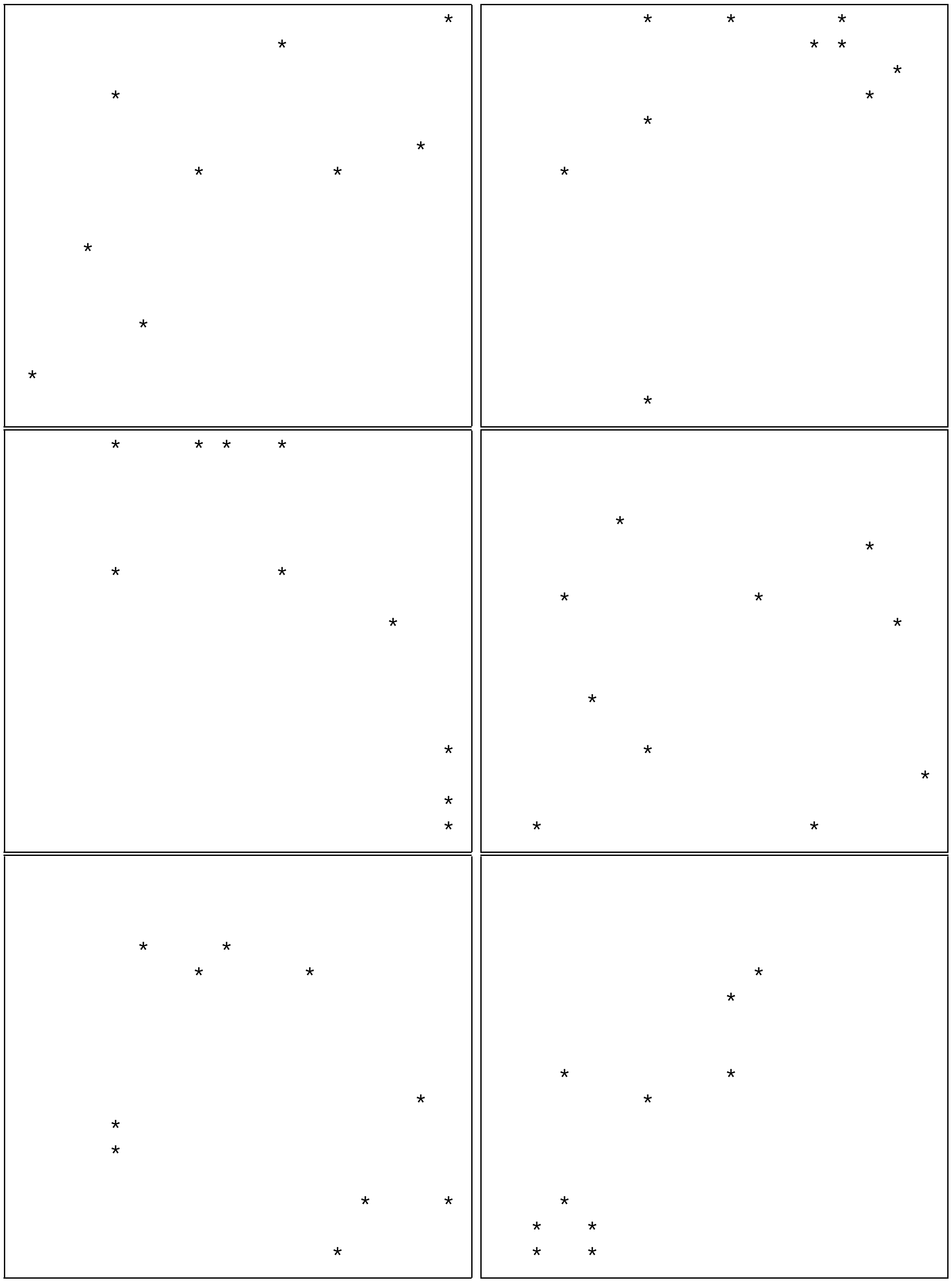


Figure 3: Torus hot spot locations cases 1, 2, 3, 4, 5, and 6.

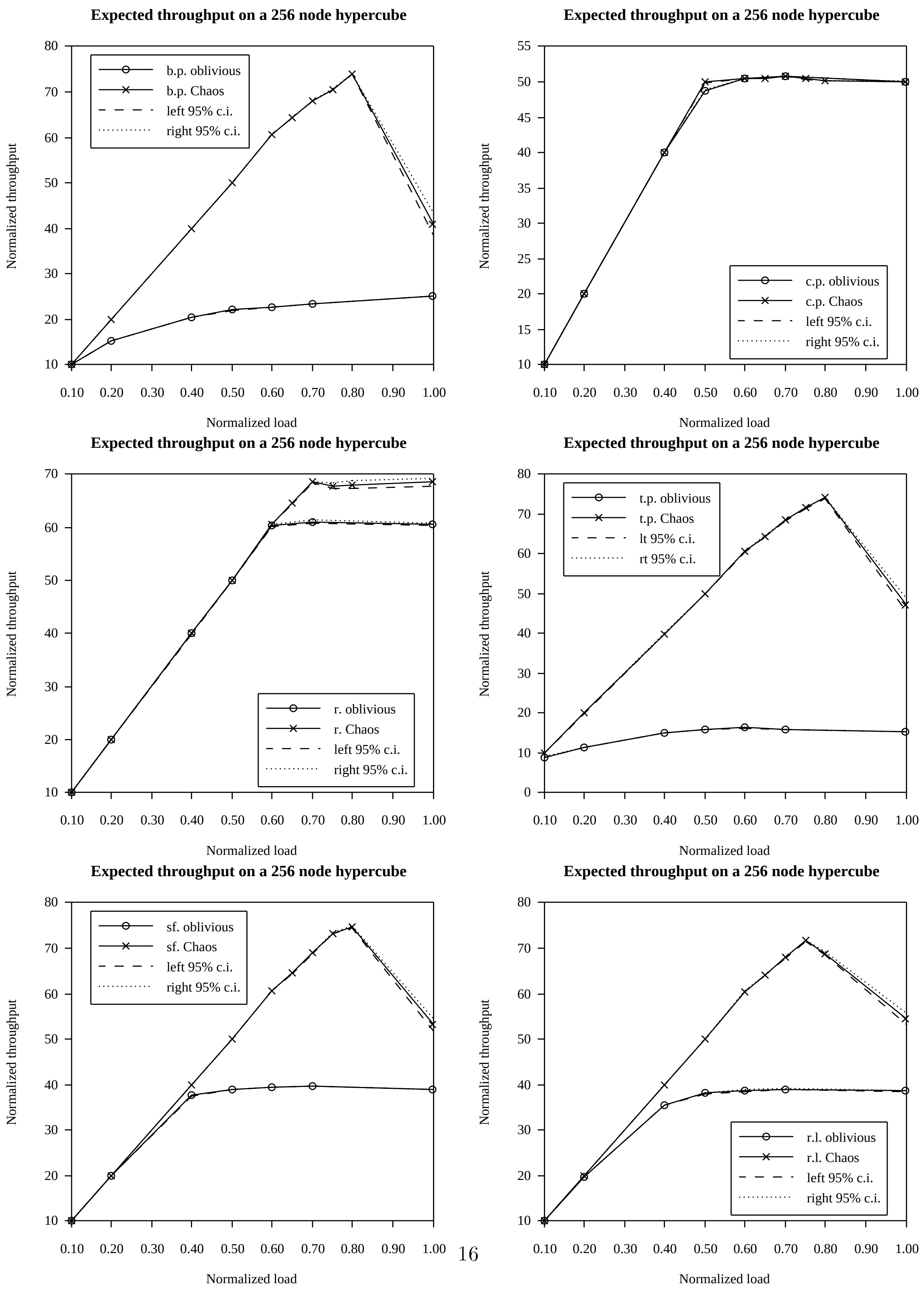


Figure 4: Hypercube throughput for bit reversal, complement, random traffic, transpose, shuffle, and random leveled traffic.

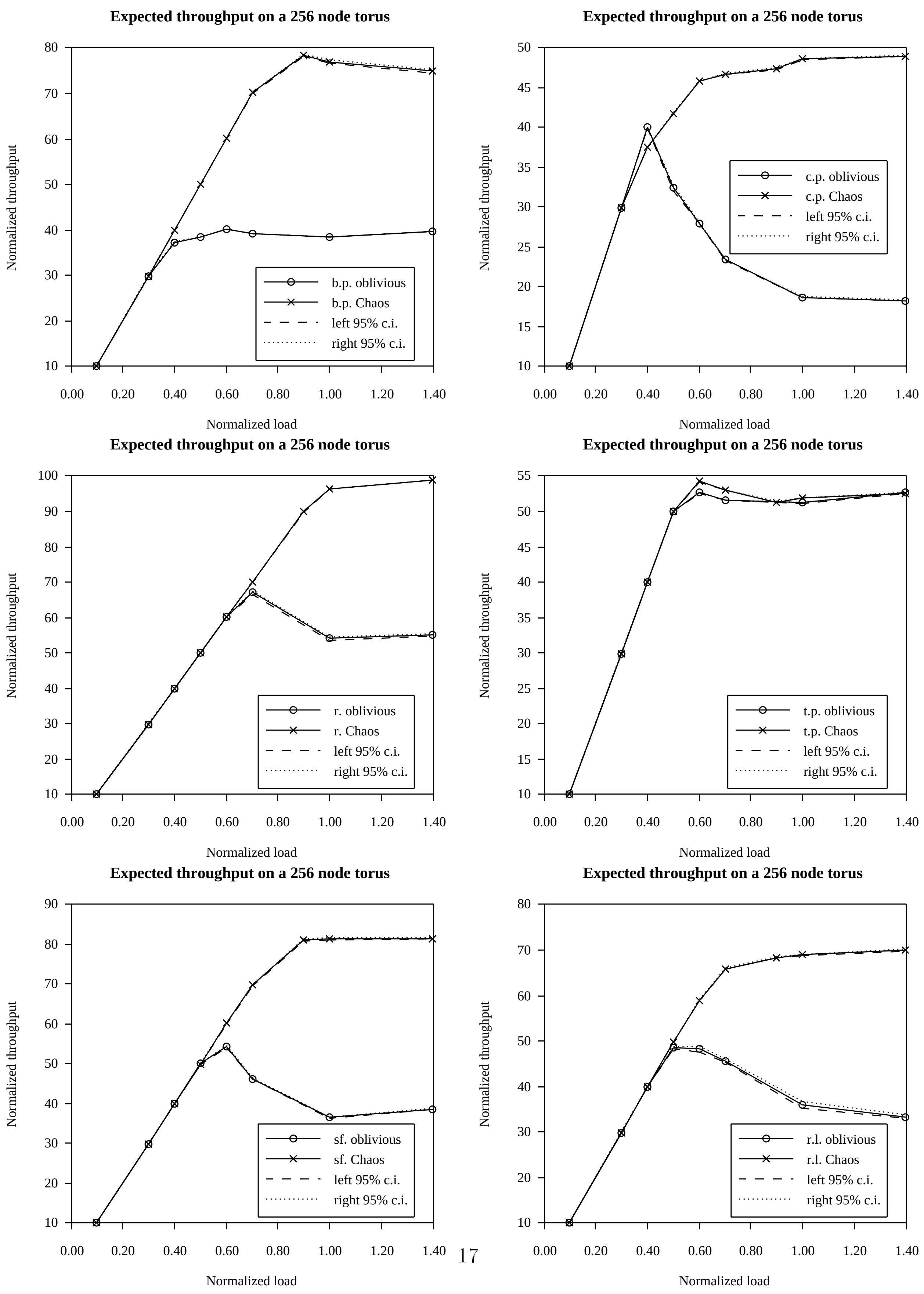


Figure 5: Torus throughput for bit reversal, complement, random traffic, transpose, shuffle, and random leveled traffic.

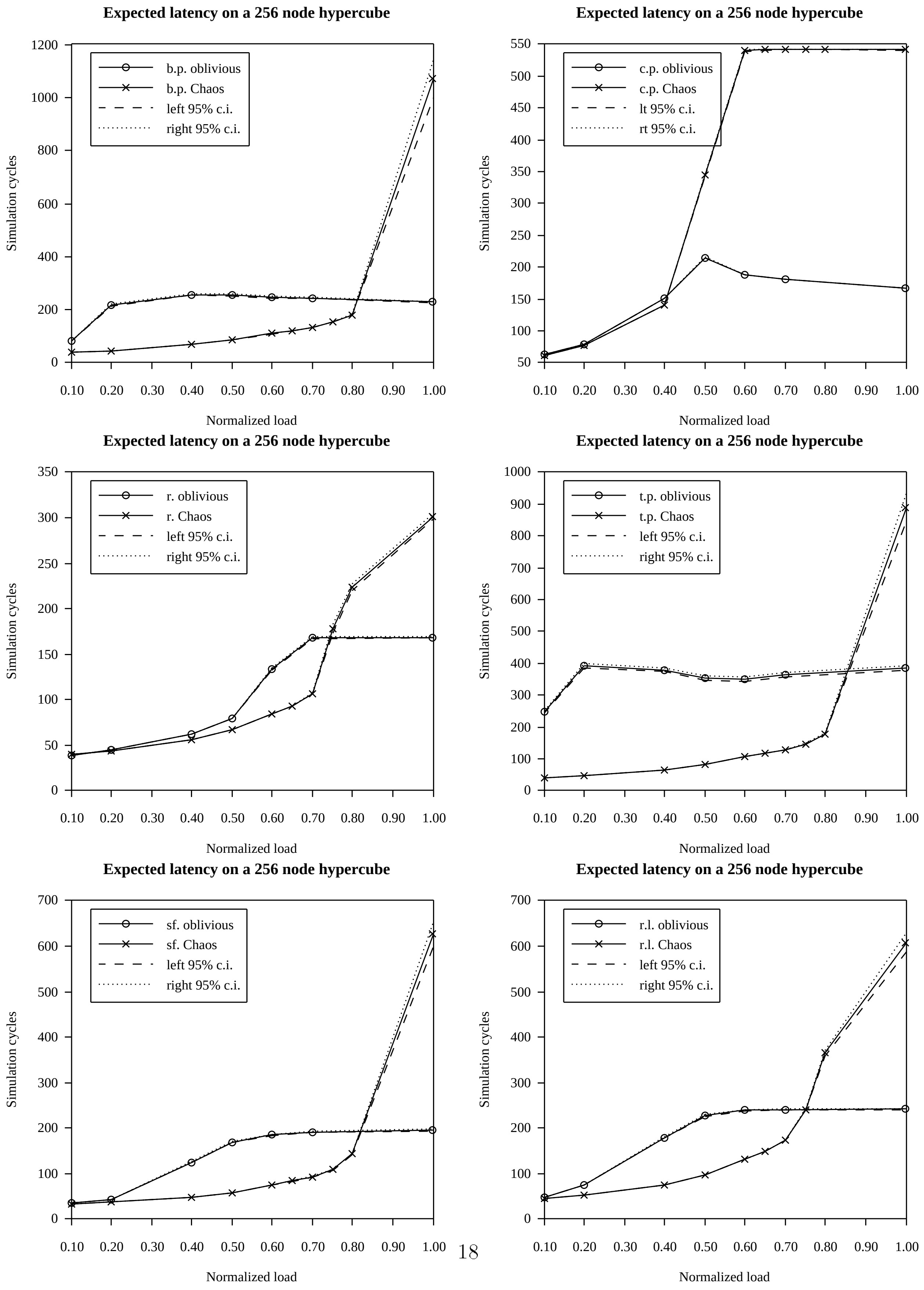


Figure 6: Hypercube latency for bit reversal, complement, random traffic, transpose, shuffle, and random leveled traffic.

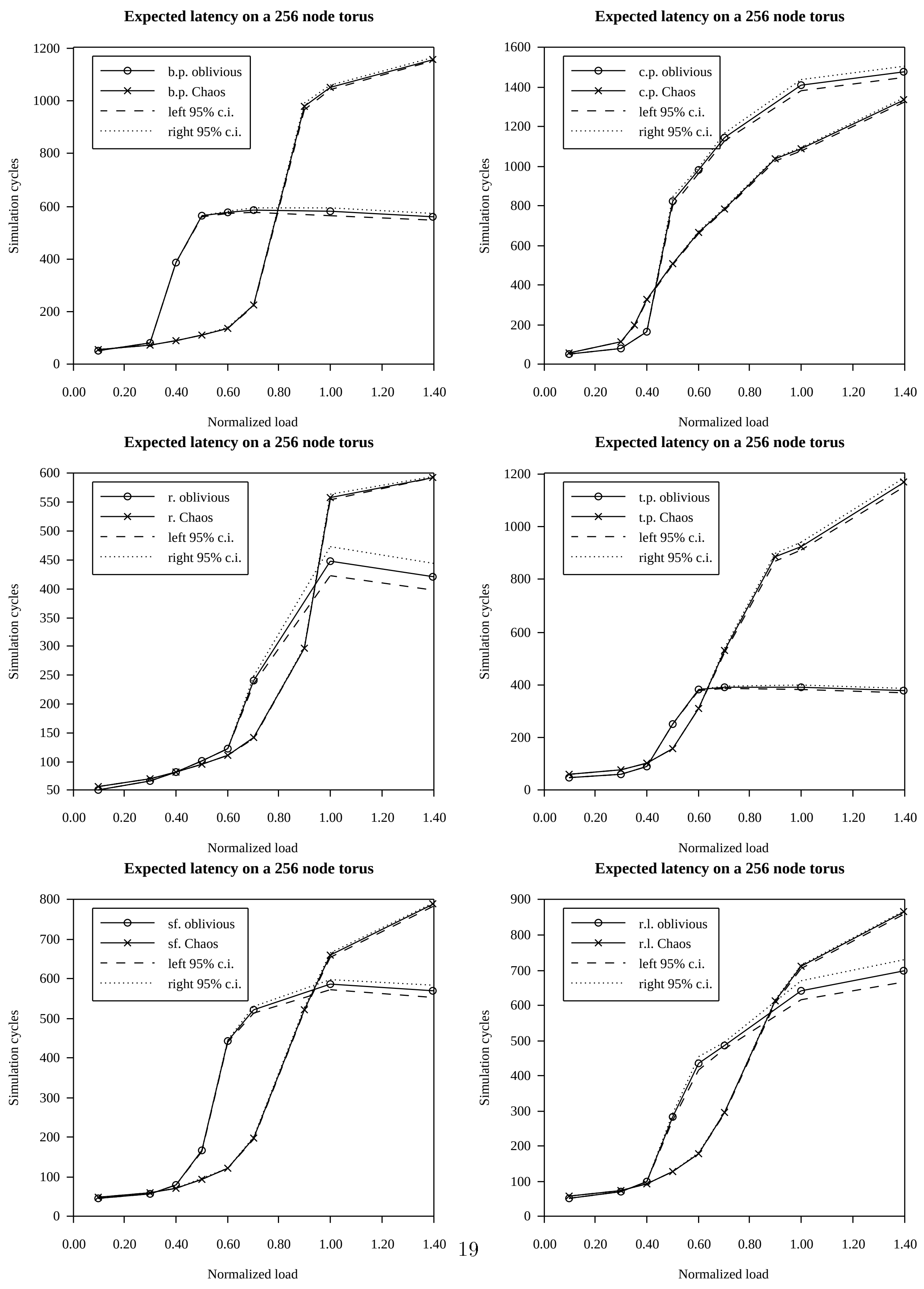


Figure 7: Torus latency for bit reversal, complement, random traffic, transpose, shuffle, and random leveled traffic.

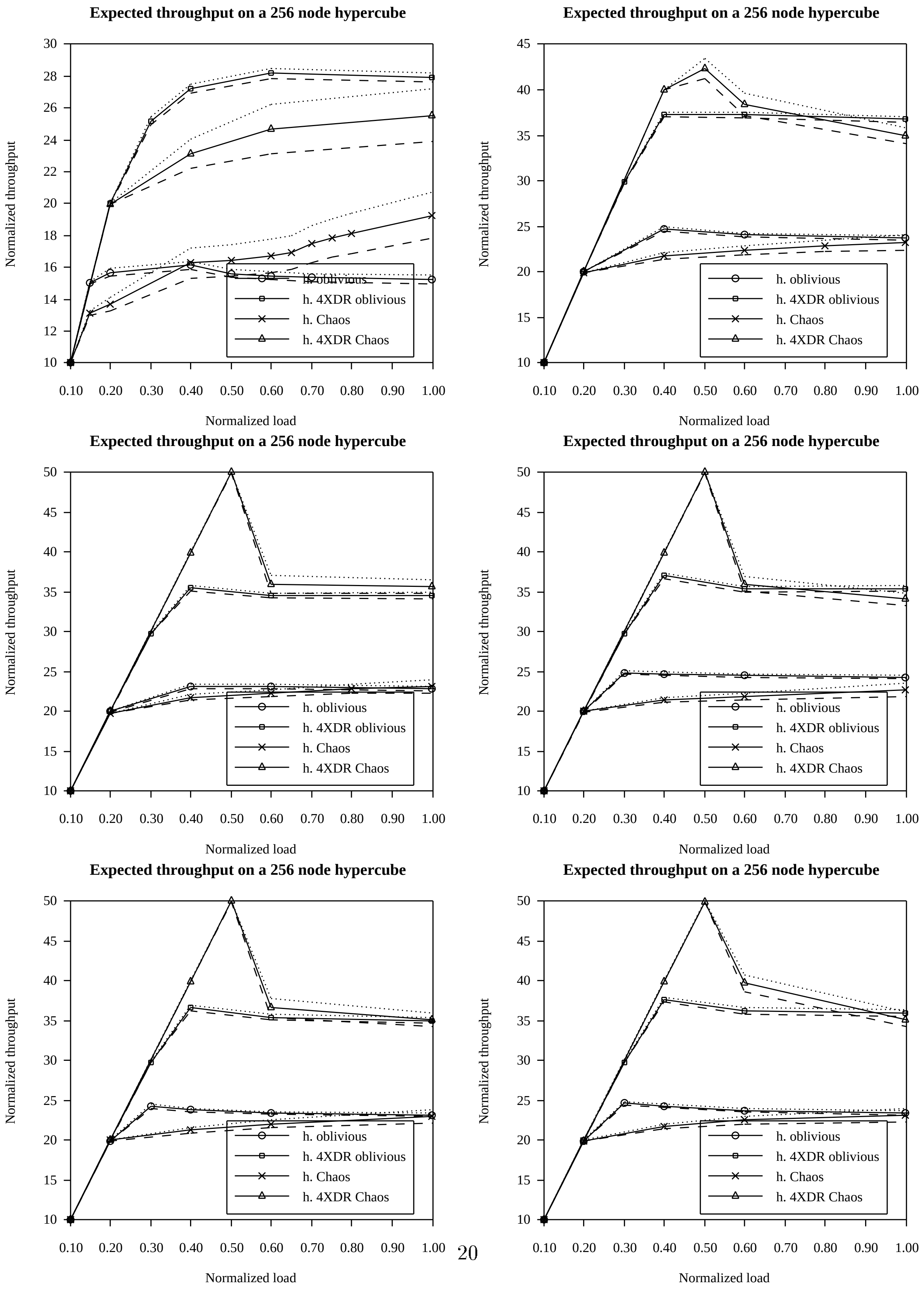
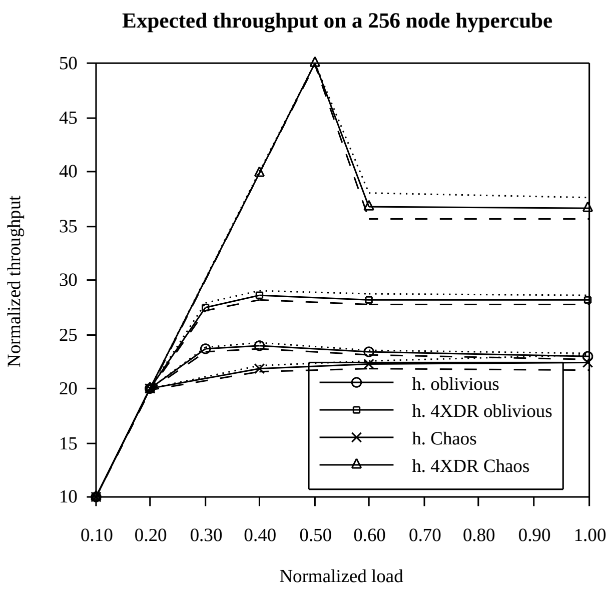


Figure 8: Hypercube hot spots throughput cases 1, 2, 3, 4, 5, and 6.



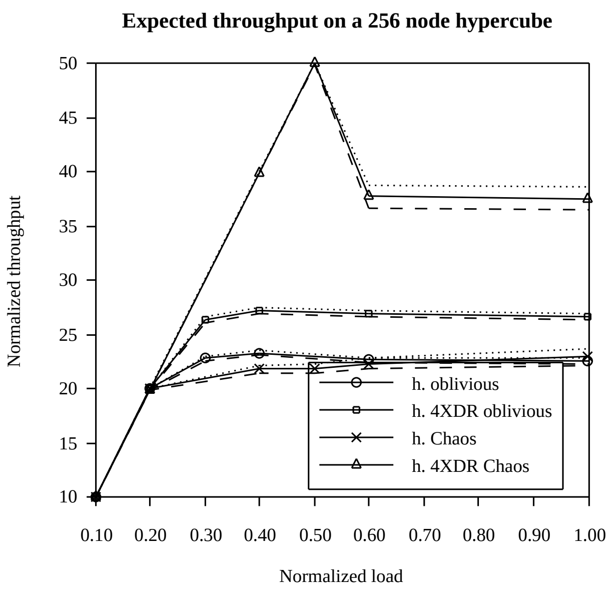


Figure 9: Hypercube hot spots throughput cases 7 and 8.

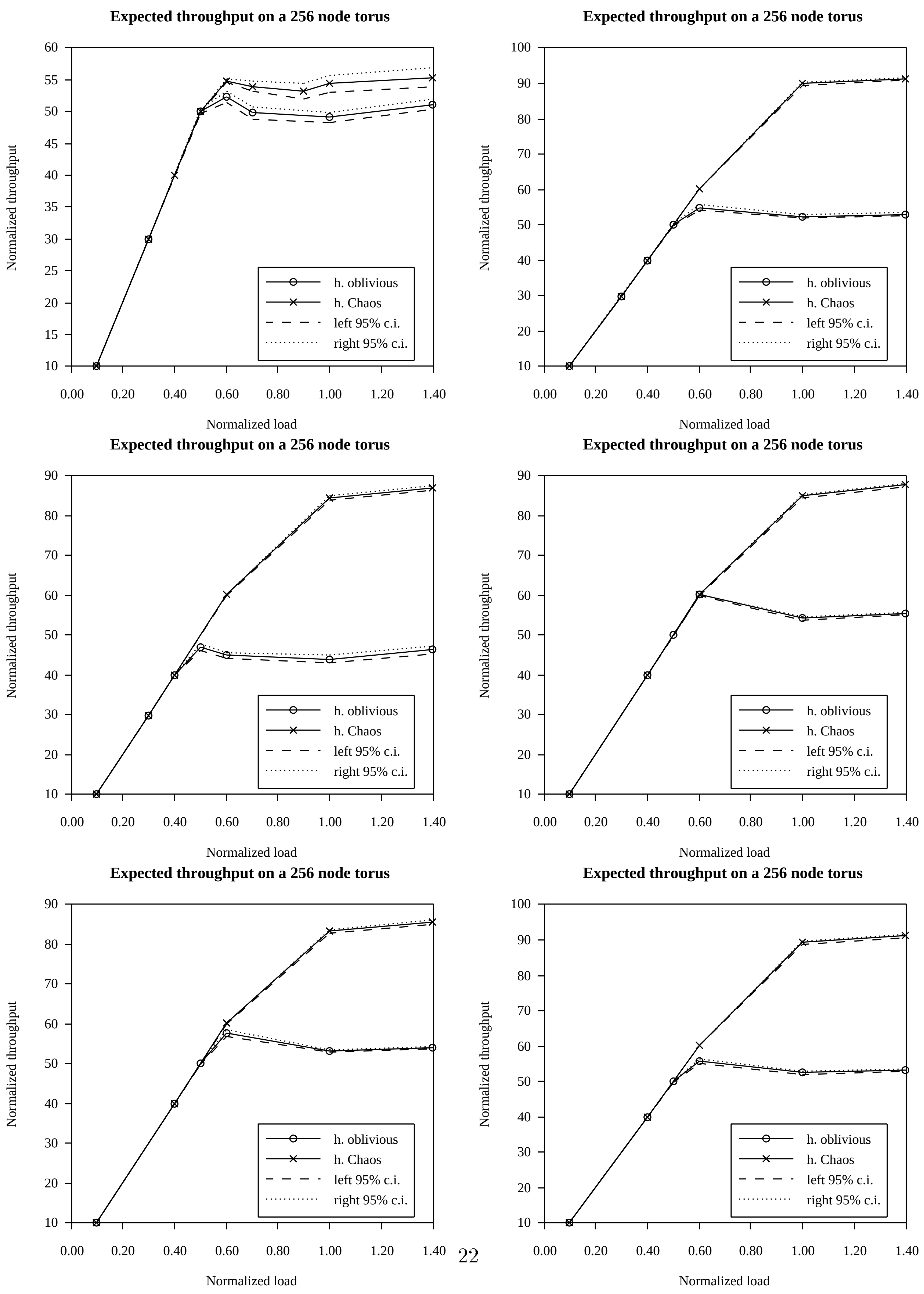


Figure 10: Torus hot spots throughput cases 1, 2, 3, 4, 5, and 6.

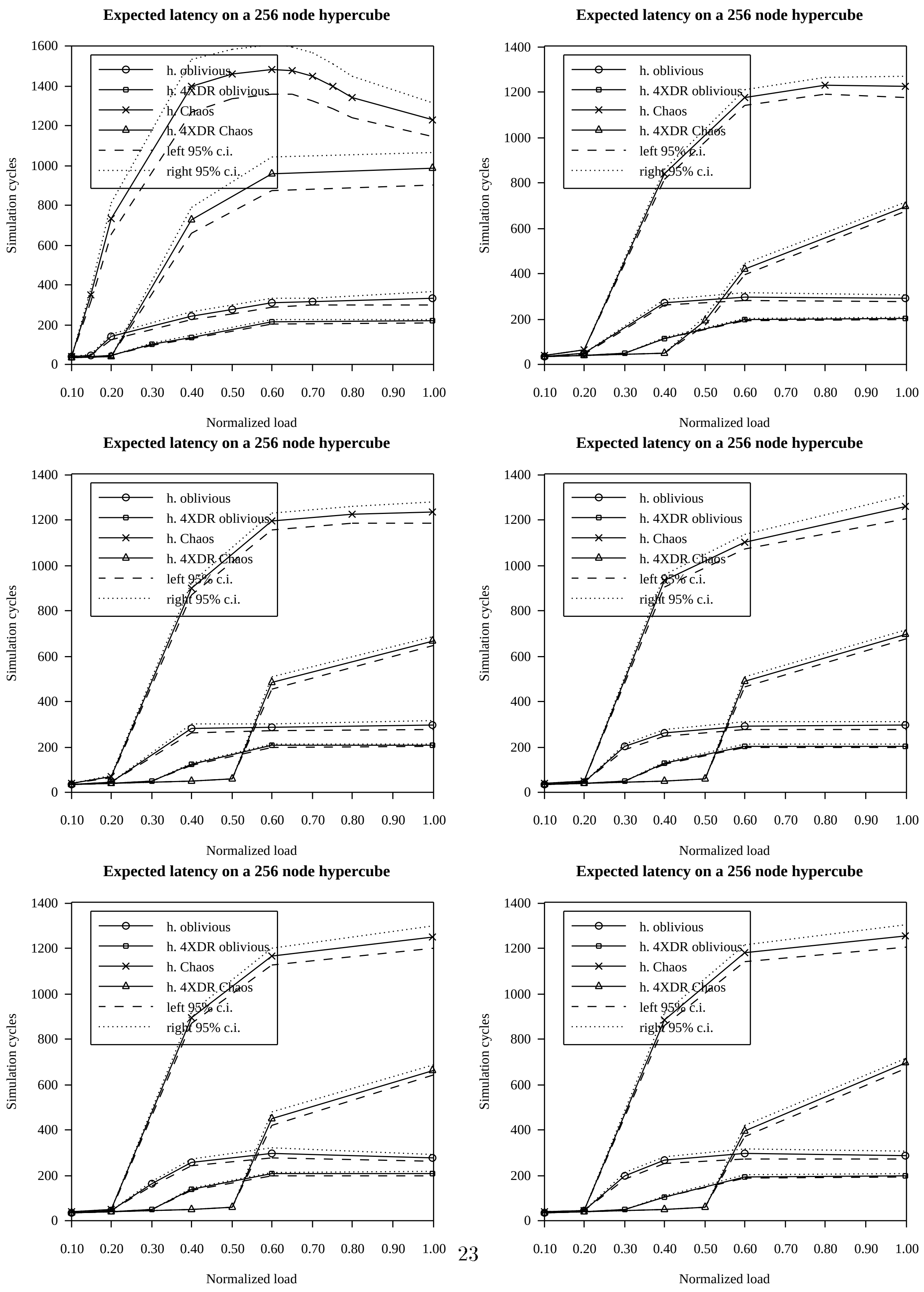
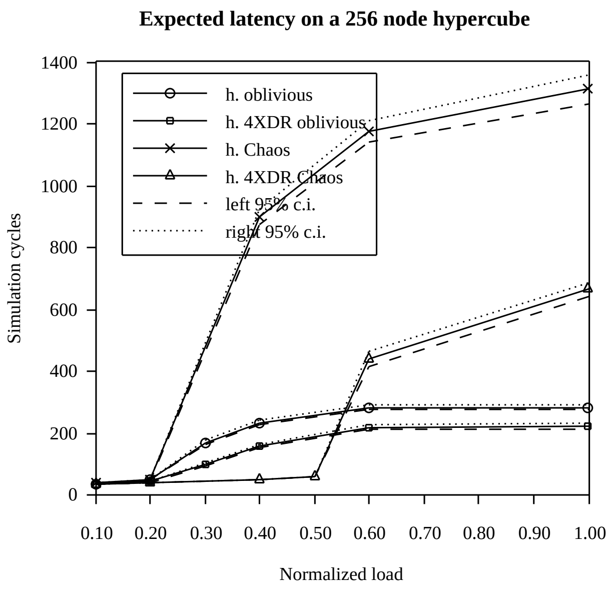


Figure 11: Hypercube hot spots latency cases 1, 2, 3, 4, 5, and 6.



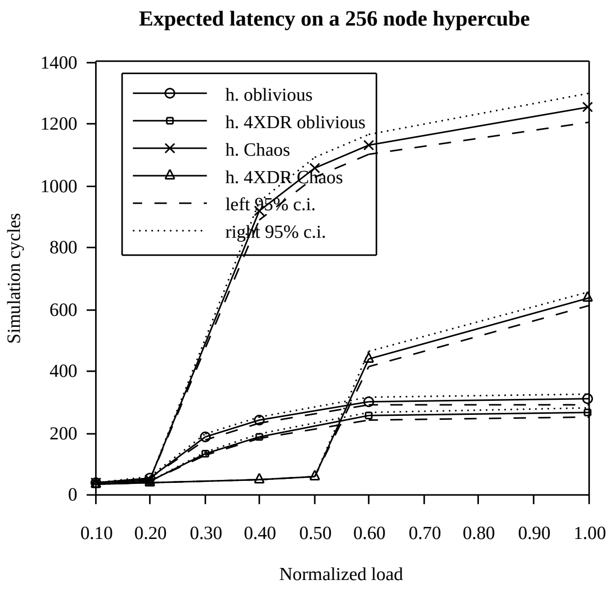


Figure 12: Hot spot latency cases 7 and 8.

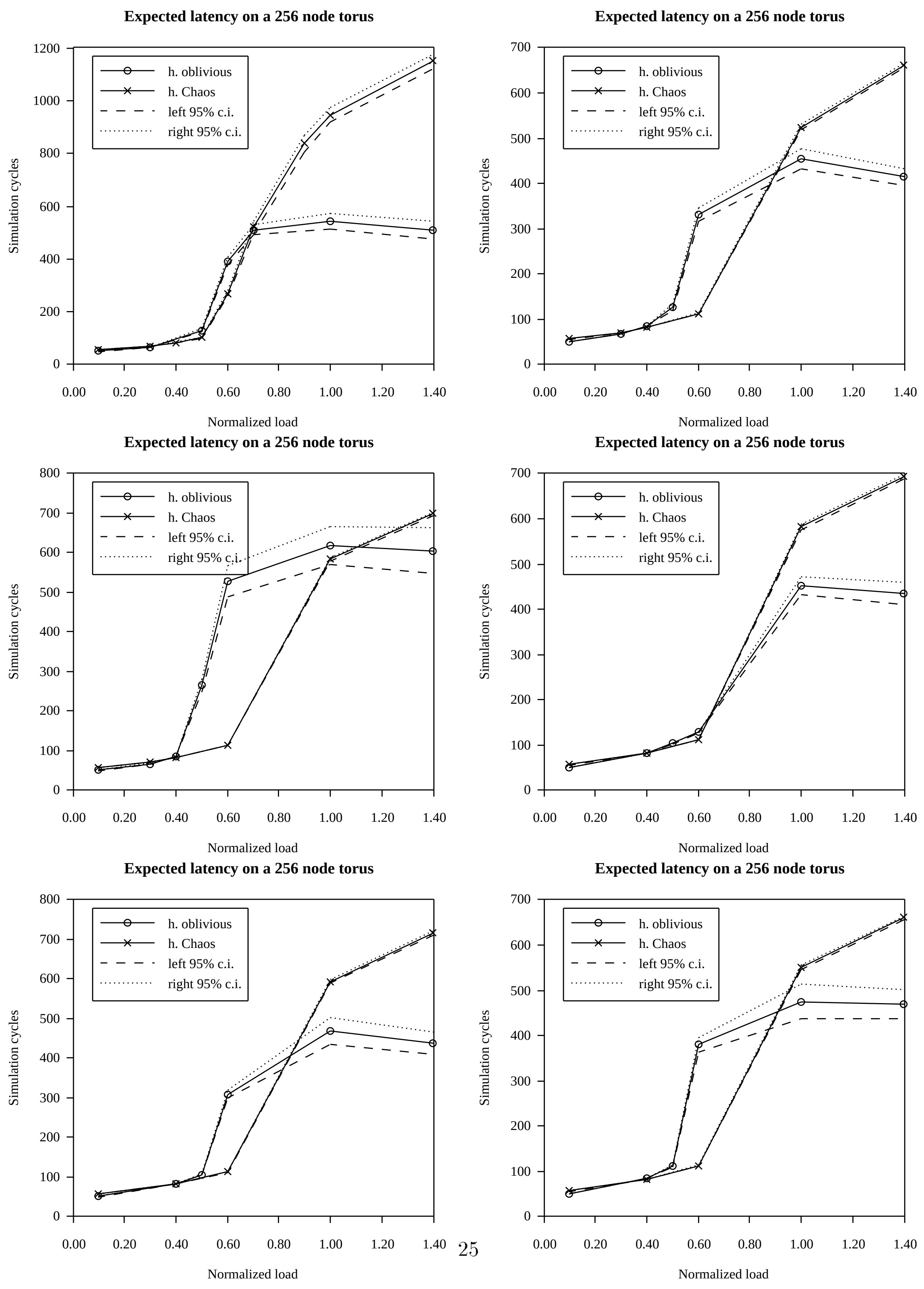


Figure 13: Torus hot spots latency cases 1, 2, 3, 4, 5, and 6.