4-1-2005

# Near-Optimal Worst-case Throughput Routing for Two-Dimensional Mesh Networks

Daeho Seo

Akif Ali

Won-Taek Lim

Nauman Rafique

Mithuna Thottethodi

# NEAR-OPTIMAL WORST-CASE THROUGHPUT ROUTING FOR TWO-DIMENSION MESH NETWORKS

DAEHO SEO
AKIF ALI
WON-TAEK LIM
NAUMAN RAFIQUE
MITHUNA THOTTETHODI

TR-ECE – 05-03
APRIL 2005

PURDUE UNIVERSITY

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, IN 47907-2035

# Near-Optimal Worst-case Throughput Routing for Two-Dimensional Mesh Networks[*]

Daeho Seo Akif Ali Won-Taek Lim Nauman Rafique

Mithuna Thottethodi

School of Electrical and Computer Engineering

Purdue University

West Lafayette, IN 47907-2035

{seod,ali2,wlim,nrafique,mithuna}@purdue.edu

**Abstract**

Minimizing latency and maximizing throughput are important goals in the design of routing algorithms for interconnection networks. Ideally, we would like a routing algorithm to (a) route packets using the minimal number of hops to reduce latency and preserve communication locality, (b) deliver good worst-case and average-case throughput and (c) enable low-complexity (and hence, low latency) router implementation. In this paper, we focus on routing algorithms for an important class of interconnection networks: two dimensional (2D) mesh networks. Existing routing algorithms for mesh networks fail to satisfy one or more of design goals mentioned above. Variously, the routing algorithms suffer from poor worst case throughput (ROMM [13], DOR [24]), poor latency due to increased packet hops (VALIANT [32]) or increased latency due to hardware complexity (minimal-adaptive [7, 31]).

[*] This report is an expanded version of a previously published paper [18].

The major contribution of this paper is the design of an oblivious routing algorithm—O1TURN—with provable near-optimal worst-case throughput, good average-case throughput, low design complexity and minimal number of network hops for 2D-mesh networks, thus satisfying all the stated design goals. O1TURN offers optimal worst-case throughput when the network radix ($k$ in a $k$x$k$ network) is even. When the network radix is odd, O1TURN is within a $1/k^2$ factor of optimal worst-case throughput. O1TURN achieves superior or comparable average-case throughput with global traffic as well as local traffic. For example, O1TURN achieves 18.8%, 0.7% and 13.6% higher average-case throughput than DOR, ROMM and VALIANT routing, respectively when averaged over one million random traffic patterns on an 8x8 network. Finally, we demonstrate that O1TURN is well suited for a partitioned router implementation that is of similar delay complexity as a simple dimension-ordered router. Our implementation incurs a marginal increase in switch arbitration delay that is completely hidden in pipelined routers as it is not on the clock-critical path.

# 1 Introduction

Two dimensional (2D) mesh networks constitute an important class of interconnection networks and they have been used in both commercial and research machines [1, 3]. There is renewed interest in the 2D-mesh topology because it is one of the natural topologies for onchip networks [6, 12, 28, 27]. Many tiled micro architectures that have been proposed rely on an underlying mesh or mesh-like networks for communication [28, 16, 26] among tiles.

Throughput and latency are the primary performance metrics to evaluate interconnection networks. Ideally, routing algorithms must aim to maximize throughput in both the worst case and the average case. A routing algorithm must also aim to minimize the following two components of latency: (a) the number of network hops, which directly depends on the routing algorithm and (b) delay through a router, which depends on complexity of the router implementation. Indirectly, this is also a function of the routing algorithm as the complexity of router implementation is often a direct outcome of the complexity of the routing algorithm. In summary, high worst-case and average-case throughput, minimal number of network hops and low complexity router implementation are desirable goals for interconnection network designers. This paper addresses the challenge of designing a routing algorithm for 2D-mesh networks that satisfies all these design goals.

Existing network designs compromise on one or more of the design goals listed above. For example, dimension-ordered routing [24] (DOR) suffers from poor worst-case and average-case throughput, especially at larger network sizes, because it offers no routing flexibility. In contrast, the routing algorithm proposed by Valiant [32] achieves optimal worst-case throughput by increasing routing flexibility but suffers from poor average case-throughput and increased network hops. A third alternative—ROMM [13]—combines the benefits of minimum number of network hops, increased routing flexibility (compared to DOR) and good average-case throughput. Unfortunately, ROMM suffers from poor worst-case throughput. This result has been demonstrated for the 2D-torus[1] topology [29] We revalidate this result for mesh networks. For example, the worst-case throughput of ROMM for a 12x12 mesh network is 5% worse than DOR and 48% less than optimal (Section 2.2). Finally, there are adaptive routing algorithms. Unlike ROMM, VALIANT and DOR, which route packets based purely on the source, destination and randomization[2], adaptive routing may factor network conditions into the routing decision. Adaptive routing, in general, suffers a latency cost when compared to simple routing techniques like DOR, ROMM and VALIANT because of more complicated router design. The only way to potentially justify the added latency cost of adaptive routing is to demonstrate a corresponding benefit by improvements in throughput. But if a simpler (i.e., non adaptive) routing algorithm exists that is capable of near-optimal worst-case throughput and good average-case throughput, there exists little-to-no opportunity for improvement by adaptive routing.

Our routing algorithm—O1TURN or *Orthogonal one-turn* routing—overcomes the various problems and design trade-offs discussed above. O1TURN allows packets to traverse one of two possible dimension-ordered routes that differ only in the order of dimension traversal (X-first Y-next OR Y-first, X-next). While O1TURN is much more restrictive than ROMM in the number of potential routes, the limited amount of routing flexibility allowed

[1] A 2D-torus is a mesh network with additional wrap-around links.

[2] This class of algorithms are called *oblivious* routing algorithms

by O1TURN is sufficient to match the average-case throughput achieved by ROMM. In addition, O1TURN offers provable, near-optimal bounds on its worst-case throughput. On the latency front, O1TURN guarantees the minimum number of network hops. The complexity of a router implementation for O1TURN is comparable to that of DOR.

We are unaware of any previous oblivious routing algorithms that guarantee near-optimal worst-case throughput for 2D mesh networks. There have been previous studies that developed near-optimal worst-case and average-case throughput for the 2D-torus topology [29, 30]. These studies have shown that it is impossible to achieve optimal worst-case throughput *and* minimal routing in the torus topology. Many routing algorithms have been proposed to achieve load balancing by exploiting non-minimal routes [22, 21, 23]. All these techniques are *fundamentally* dependent on the torus topology as the routing algorithms rely on the wrap-around links to balance load. As such, they are not applicable for two-dimensional meshes.

PFNF is a minimal adaptive routing algorithm that resembles O1TURN in that it combines two types of routing techniques to balance load [31]. In contrast to O1TURN, PFNF uses adaptive routing (Positive-first and Negative-first “Turn model” routing [8]) in each of its routing layers to maximize adaptivity. Since PFNF is an adaptive routing algorithm, it suffers from the complexity and delay cost as described earlier in this section.

The main contributions of this paper are:

• We develop O1TURN a minimal, oblivious routing algorithm that achieves near-optimal worst-case throughput for two-dimensional mesh networks. It is optimal for all $k$x$k$ mesh networks with even $k$. When $k$ is odd, O1TURN is within $1/k^2$ of the optimal worst-case throughput. The quadratic term in the denominator expedites the convergence of the worst-case throughput of O1TURN to optimal at higher values of $k$. O1TURN also achieves average-case throughput similar to ROMM and 14% and 19% higher than VALIANT and DOR, respectively, when averaged over one million random communication patterns for an 8x8 mesh network.

• We describe a router implementation for O1TURN that is of comparable complexity as a simple DOR router. Since DOR routers are among the simplest routers in terms of complexity/delay, they are worthy designs to compare against. Evaluating the O1TURN router implementation using router delay models reveals that our implementation reduces the complexity of the critical path in the router and marginally increases the delay along a non-critical path. At worst, the marginal increase in the non-critical path is completely hidden in pipelined routers. At best, it can actually reduce the delay below that of a DOR router because of the reduction in the critical path. The same conclusions are true when O1TURN routing used in router designs with sophisticated router latency hiding techniques like speculation [15] and pre-computation [12]

The rest of this paper is organized as follows: Section 2 provides a brief background on the analytical methods for network and router model we use. Section 3 describes the O1TURN routing algorithm and analyzes its worst-case and average-case behavior. Section 4 proposes a partitioned router implementation for O1TURN that minimizes delay. Section 5 describes the evaluation methodology. We present simulation results in Section 6. Section 7 discusses related work. Section 8 summarizes and concludes this paper.

## 2 Background

In this section, we first define the terminology relevant to 2D-mesh networks. The rest of this section following the terminology is organized as follows: Section 2.1 describes various routing algorithms, and compares and contrasts them with O1TURN. Section 2.2 provides a background on the analytical methods we employ to evaluate our routing algorithm.

We define a *traffic pattern* or *permutation* as a mapping of source nodes to destination nodes. Minimal routing algorithms ensure that the packet gets closer to the destination with each network hop. The minimum rectangle is defined as the submesh that contains the source and destination nodes at its diagonally opposite corners. *Oblivious routing algorithms* are those that are oblivious of network state when determining a route. Note, oblivious algorithms may be randomized. In contrast, the routing decision in adaptive routing algorithms may also depend on network state.

## 2.1 Routing Algorithms

In this section, we describe some popular routing algorithms and the pros and cons of each design in comparison with O1TURN routing. Dimension-ordered routing (DOR) is an extremely simple routing algorithm for the broad class of networks that include 2D mesh networks. Packets simply route along one dimension first and then in the next dimension. This simplicity comes at the cost of poor worst-case and average-case throughput for mesh networks. However, its simplicity is also its strength as it enables low complexity implementations. DOR also enables switch optimizations [28, 14, 5, 10] to simplify circuitry. (Switch optimizations have also been proposed for deadlock-recovery- based adaptive routers [2], but those optimizations are not directly applicable in the context of our router model. We discuss this issue further in Section 7.) In spite of a number of critiques pointing out the shortcomings of DOR, it continues to be used and adopted. For example, MIT's RAW uses DOR in its dynamic network [28]. O1TURN is comparable in simplicity to DOR *and* it overcomes the known limitations of DOR with respect to worst-case and average-case throughput.

ROMM addresses some of the shortcomings of DOR. ROMM randomly chooses an intermediate node within the minimum rectangle defined by the source and destination nodes and routes packets via the intermediate node. The two phases[3] of routing (i.e., from source node to intermediate node and from intermediate node to destination node) may use some variation of DOR (i.e., XY-order or YX-order). While ROMM has been shown to work well for many "hard" traffic patterns, recent work has demonstrated that in the worst case, ROMM may saturate at a lower throughput than DOR in 2D-torus networks [29]. We revalidate this result for 2D mesh networks as well. ROMM remains an attractive design point for domains where average case behavior is important (such as multiprocessor interconnection networks) and worst-case throughput is less important. In comparison, O1TURN matches the average case behavior of ROMM for both global and local traffic.

Valiant proposed a routing algorithm that randomly chooses a node in the network and routes via that node [32].

[3] ROMM is actually a family of routing algorithms that can route in multiple phases. We consider the two-phase version of ROMM in this paper.

ROMM is similar to VALIANT as far as the two-phase routing is concerned. But ROMM chooses the intermediate node from within the minimal rectangle whereas VALIANT may choose an intermediate node from anywhere within the network. Consequently, VALIANT is a non-minimal routing algorithm. Though VALIANT achieves optimal worst-case throughput, it is not an attractive design point as it sacrifices average case behavior and latency (due to non-minimal routing).

DOR, ROMM and VALIANT are all oblivious routing algorithms. Adaptive routing offers routers the flexibility to react to network conditions [17, 7, 8, 31, 11]. As discussed earlier, adaptive routing suffers a latency cost when compared to simple routing techniques like DOR, ROMM and VALIANT because of more complicated router design. Further, the existence of oblivious routing algorithms that achieve near-optimal worst-case throughput implies that there is no benefit corresponding to the latency cost.

## 2.2 Throughput Analysis of Oblivious Routing Algorithms

In this section, we provide a brief overview of analytical methods used to evaluate ideal throughput. In particular, we elaborate on the concept of network capacity and on a method to compute worst-case throughput for oblivious algorithms. To simplify the discussion on throughput analysis, we assume that one packet is made up of exactly one flit and that it can route from node to node in one cycle. This assumption greatly simplifies the explanation by eliminating extraneous issues related to flow control. We explain the effects of adding realistic flow control later. Also, all our results use realistic (i.e., multiple-flit) packet sizes for evaluation of O1TURN.

**Network capacity** Network capacity is defined as the maximum sustainable throughput when a network is loaded with uniform random traffic. An expression for network capacity ($N_c$ packets/node/cycle) can be derived by computing the load at which the network bisection links are fully utilized. Consider a network bisection that divides a $k$x$k$ network into two parts: one with $k.\lfloor\frac{k}{2}\rfloor$ nodes and another with $k.\lceil\frac{k}{2}\rceil$ nodes. There are a total of $k$ channels that connect the two parts in each direction. Equating the packets generated in one half that are likely to cross the bisection with the number of packets that can cross the bisection when the bisection links are 100%

utilized, we get the following expression.

$N_c.(k.\lfloor\frac{k}{2}\rfloor).(k.\lceil\frac{k}{2}\rceil/k^2)) = k$ i.e., $N_c = k/(\lfloor\frac{k}{2}\rfloor.\lceil\frac{k}{2}\rceil)$

When $k$ is even, the expression simplifies to $N_c = (4/k)$ packets/node/cycle. The term simplifies to $N_c = 4.k/(k^2 - 1)$ packets/node/cycle when $k$ is odd.

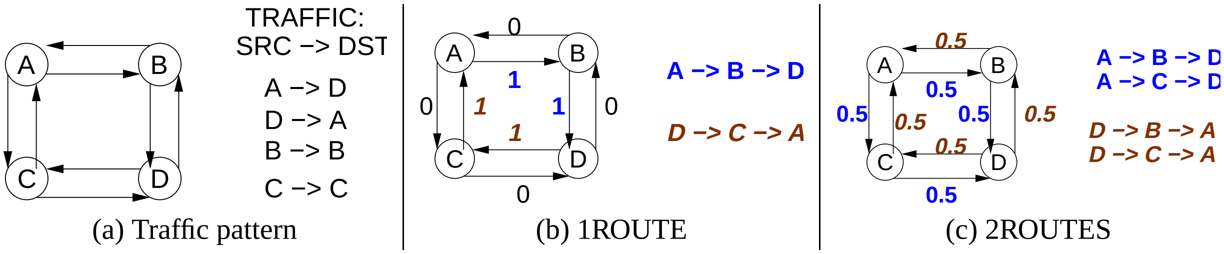
This coarse-grain analysis is sufficient to determine network capacity. However, analyzing the worst-case and average-case throughput of routing algorithms requires finer analytical tools. In the rest of this section, we give a brief background into the techniques to compute worst-case and average-case throughput of oblivious-routing algorithms.

**Worst-case and Average-case throughput analysis** Towles and Dally [29] developed a methodology to analytically compute the ideal worst-case throughput of oblivious routing algorithms. The technique developed therein, which we refer to as the *TD-method* in the rest of this paper, provides interesting results that at first glance appear counter-intuitive. For example, the TD-method demonstrated that the worst-case throughput of ROMM was actually worse than that of DOR. Our analysis of O1TURN relies on the TD-method. As such, we offer a brief background of the technique without delving into the details.

The key innovation of the technique was a method to construct an adversarial traffic pattern that causes links to saturate. This is achieved by computing the *channel-load*. The channel with the highest load is the bottleneck channel in the whole network and it determines the saturation throughput of the network as a whole. The challenge of finding the worst-case throughput for a routing algorithm reduces to finding the channel with the highest load.

The next paragraph describes how channel load is computed by the TD-method. We refer the reader to the original paper for details on how the TD-method computes adversarial traffic patterns for oblivious routing algorithms [29].

Figure 1 illustrates the channel load computation with a simple example. For a 2x2 network and traffic pattern specified in Figure 1(a), we wish to find the channel loads for two different routing algorithms. The first routing algorithm—1ROUTE—allows packets to traverse only one route as shown in Figure 1(b). This contributes a



(a) Traffic pattern (b) 1ROUTE (c) 2ROUTES

Figure 1: Channel-load Computation Example

channel load of 1 on the channels along each packet's respective path. The maximum channel load in this case is 1. The reciprocal of the maximum channel load (=1/1 = 1 packet/node/cycle) is the ideal saturation throughput for that traffic pattern.

The second routing algorithm–2ROUTES—routes packets in one of two routes with equal probability as shown in Figure 1(c). Since the packets from the source nodes are equally distributed on two possible paths, the channel load contribution from the packet amounts to 0.5 along each of the two paths. In this case, the ideal saturation throughput is $1/0.5 = 2$ packets/node/cycle.

## 3 The O1TURN Routing Algorithm

O1TURN routing—*orthogonal one-turn routing*— permits packets to turn at-most once (hence, the name "one-turn") during its network traversal. We add the term "orthogonal" to the name to disallow 'U'-turns (i.e., turns that reverse direction along the same dimension). This is equivalent to saying that all allowed routes must be minimal. O1TURN allows each packet to traverse one of at most two routes with equal probability. In a 2D mesh, there are at-most two minimal, one-turn routes between any given source-destination pair. The two routes differ in the order of dimension traversal (XY or YX). O1TURN chooses first dimension of traversal randomly.

Note, it is possible to describe O1TURN as a restricted version of ROMM routing with the intermediate node being one of four corners of the minimum rectangle. It may appear counter-intuitive that a less flexible routing algorithm (O1TURN) achieves better worst-case throughput than a more flexible routing algorithm (ROMM), but the restriction is fundamental to achieve near-optimal worst-case throughput. The reason for ROMM's degenerate behavior in the worst-case is the fact that with ROMM routing, communication between a source-destination pair can contribute to channel load in rows/columns of the mesh other than those that contain the source and destination nodes. This is because ROMM may contribute to channel load in the row and column of the intermediate node.

Restricting the intermediate node to be in the corner of the minimal rectangle eliminates this condition. As a result, O1TURN ensures that communication between any source-destination pair can *only* contribute to channel load in their respective rows and/or columns. This condition is sufficient to prove the near-optimal behavior of O1TURN and the proof is presented in the rest of this section.

### 3.1 Worst-case Throughput Analysis

In this section, we will prove that O1TURN is an optimal worst-case throughput when the network radix ($k$ in a $k$x$k$ network) is even. When $k$ is odd, O1TURN is within a factor of $1/k^2$ of the optimal worst case. Since the $(1/k^2)$ term diminishes quadratically with $k$, O1TURN quickly converges closer to the optimal throughput as $k$ increases. For example, O1TURN is within 1.25% of optimal for a 9x9 mesh.

We prove the above claims in two steps. First, we prove that the worst case channel load in a $k$x$k$ network with O1TURN routing is upper-bounded by $(k/2)$. This implies that the worst-case throughput of O1TURN is $(k/2)^{-1} = (2/k)$. Next, using the expression for network capacity described in Section 2.2, we characterize the near optimal nature of O1TURN.

**Claim 1** *The maximum channel load in a $k$x$k$ network with O1TURN routing is $(k/2)$.*

**Proof** Consider a channel $C$ that originates from the node $n$ at $(x_n, y_n)$ and links to a neighboring node in the positive X direction. We define the channel load on $C$ as the sum of two components – the source component and the destination component.

The source component of the channel load on $C$ is the load that originates from source nodes that belong to the set $S_1 = \{(x_s, y_s)|x_s \leq x_n, y_s = y_n\}$ and terminates at destination nodes from the set $D_1 = \{(x_d, y_d)|x_d > x_n\}$, as shown in Figure 2(a). In other words, every packet that goes from source nodes that are on the left of $C$ with the same Y co-ordinate as $n$ to destination nodes that are to the right of $n$ in Figure 2(a) contributes some load to channel $C$. This is clearly true because packets that traverse dimensions in X-Y order have to cross channel $C$. The number of source nodes that can contribute to channel load on $C$ using XY routing is at most $|S_1| = x_n$. We account for a channel load of $(1/2)$ from each source node because O1TURN lets packets traverse dimensions in the X-Y order with a probability of 50%. Thus the source component of aggregate channel load is $(x_n/2)$.

Note, if the destination shares the same Y co-ordinate as the source ($y_n$), that particular source-destination pair actually contributes a channel load of 1 instead of $(1/2)$ because there is only one minimal path between the two nodes. But we account for that channel load of 1 as equally split between source and destination nodes. This is explained in the next paragraph where we discuss the destination component.

The destination component of the channel load on $C$ arises when packets are sent from source nodes in the set $S_2 = \{(x_s, y_s)|x_s \leq x_n\}$ to destination nodes in the set $D_2 = \{(x_d, y_d)|x_d > x_n, y_d = y_n\}$, as shown in Figure 2(b). In this case, packets going from nodes to the left of $C$ to nodes that are on the right of $n$ with the same Y co-ordinate as $n$ contribute to the channel load on $C$ when the packets route in the Y-X dimension order. There can be at most $|D_2| = k - x_n$ source-destination pairs that satisfy this condition. Again, since O1TURN imposes a 50% probability of traversing dimensions in the Y-X order, each such source-destination pair contributes a load of $(1/2)$ to the channel load. The total contribution of the “destination component” of channel load is $(k - x_n)/2$.

The accounting of “destination component” as described above, also clarifies the accounting of channel load when both source and destination nodes are on the same row as the channel $C$ (i.e., $y_s = y_d = y_n$). The channel load of 1 is accounted for by counting the source component $(1/2)$ separately from the destination component $(1/2)$.

The maximum total channel load ($c_{max}$, is the sum of the two components of channel load. Thus, $c_{max} = (x_n/2) + (k - x_n)/2 = k/2$.

This analysis is symmetrical for every dimension and direction. Thus the maximum channel load for a $k$x$k$ network with O1TURN routing is $(k/2)$.

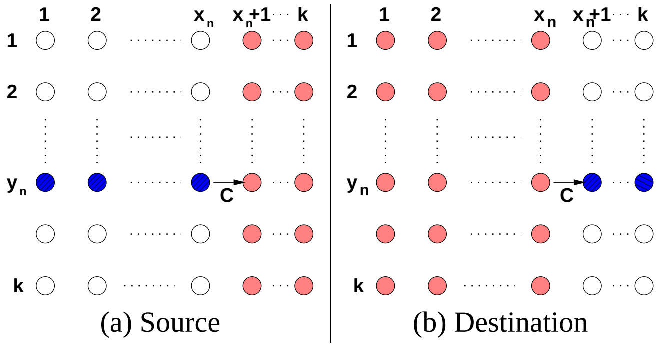
Since the maximum channel load is shown to be $(k/2)$, the absolute worst-case throughput is given by $(k/2)^{-1} = 2/k$. The optimal worst case throughput ($N_{wc-opt}$) is known, from previous literature, to be 50% network capacity [29]. Recall, the network capacity is $(4/k)$ for even $k$ and $4k/(k^2 - 1)$ when $k$ is odd (see Section 2.2). From this we get, $N_{wc-O1TURN} = (2/k) = (1/2).(4/k) = N_{wc-opt}$ when $k$ is even. For odd $k$ , $N_{wc-O1TURN} = (2/k) = (1/2).((4k/(k^2 - 1)).(k^2 - 1)/k^2 = N_{wc-opt}.(1 - 1/k^2)$.

Thus O1TURN is optimal for all even $k$ and within a $(1/k^2)$ term for odd $k$. Figure 3 plots the worst-case throughput of O1TURN, DOR and ROMM in comparison with optimal for values of $k$ between 2 and 16. The numbers for ROMM are generated using the TD-method [29]. The curve illustrates two important points: (a) Not only is O1TURN better in the worst-case than ROMM and DOR, its trend as $k$ grows is also superior. O1TURN approaches the optimal worst-case throughput as $k$ grows whereas ROMM and DOR diverge and become worse with increasing $k$. (b) ROMM achieves better worst-case throughput than DOR at low values of $k$, but there is a crossover point at $k = 10$ beyond which ROMM is worse than DOR.

Note, the absolute optimal worst-case throughput degrades as the network radix increases. While this is true for global traffic, local traffic effectively divides the larger mesh into smaller submeshes and is able to achieve higher throughput. O1TURN works well for local traffic as we show later in this section because it achieves worst-case optimal routing in smaller submeshes.

**Average-case Throughput:**

In an extension of the TD-method analysis, Towles *et.al.* show that average case behavior can be closely approximated by computing the harmonic mean of worst-case throughputs for a sample of random communication patterns [30]. We adopt the same technique and evaluate the average-case throughput of O1TURN with a sample



(a) Source

(b) Destination

Figure 2: Components of channel load for O1TURN

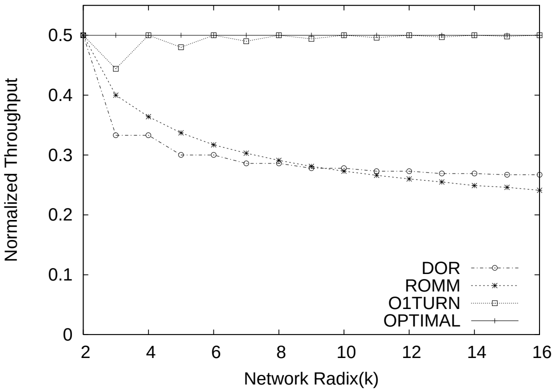
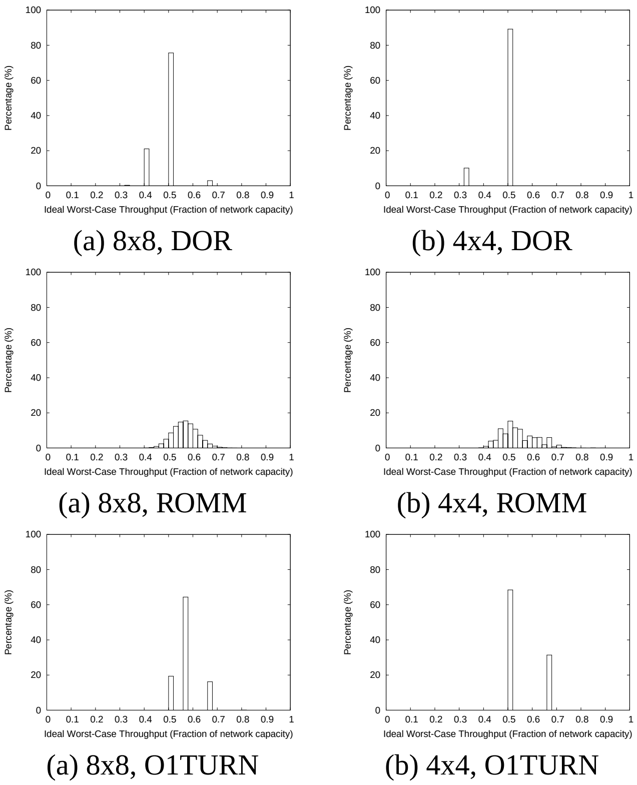


Figure 3: Worst-case Channel Load

size of one million random permutations. The average-case throughputs for 8x8 and 4x4 networks are tabulated in Table 1. (The table also includes the saturation throughputs of the four oblivious routing algorithms under various traffic patterns. The worst case traffic for each routing algorithm is different and is obtained using the TD-method.)

The average case throughputs of O1TURN and ROMM were within 2% of each other. Since the average case analysis is an approximate computation, the 2% difference is not meaningful. At best, we may conclude that the average case throughputs of ROMM and O1TURN are comparable. Both O1TURN and ROMM achieve between 11% and 19% better average-case throughput compared to DOR. The distribution of normalized worst-case throughputs for one million random permutations is presented in the histogram in Figure 4 for two network sizes. Note, O1TURN does not have a single instance with throughput less than 50% of network capacity, as



(a) 8x8, DOR (b) 4x4, DOR

(a) 8x8, ROMM (b) 4x4, ROMM

(a) 8x8, O1TURN (b) 4x4, O1TURN

Figure 4: Worst-case throughput distribution for one million random permutations

expected. The histograms of DOR and O1TURN show discrete clusters because channel loads are discrete amounts in the two routing algorithms. We see a smoother distribution for ROMM because the varying probability of choosing intermediate nodes introduces a larger range of values for channel load.

**Average Aggregate Saturation Throughput for Local Traffic:**

Many practical applications of mesh networks rely on purely local communication within a submesh. For example, if the threads of an application are mapped to a subset of processors, any network traffic generated by those threads will remain constrained in the nodes and links within the bounding rectangle of those processors.

To evaluate the average case throughput of O1TURN under local traffic, we divide an 8x8 network into multiple submeshes. We then analyze the average case throughput of the network assuming network traffic is local to each submesh, i.e., each packet's source and destination nodes are contained within the same submesh. If the traffic

|  | **8x8 Network** |  |  |  |
|---|---|---|---|---|
|  | VALIANT | ROMM | DOR | O1TURN |
| Random | 0.5 | 1 | 1 | 1 |
| Transpose | 0.5 | 0.814 | 0.286 | 0.572 |
| Complement | 0.5 | 0.324 | 0.5 | 0.5 |
| Perf. shuffle | 0.5 | 0.708 | 0.5 | 0.667 |
| Worst-case | 0.5 | 0.292 | 0.286 | 0.5 |
| Avg. case | 0.5 | 0.564 | 0.478 | 0.568 |

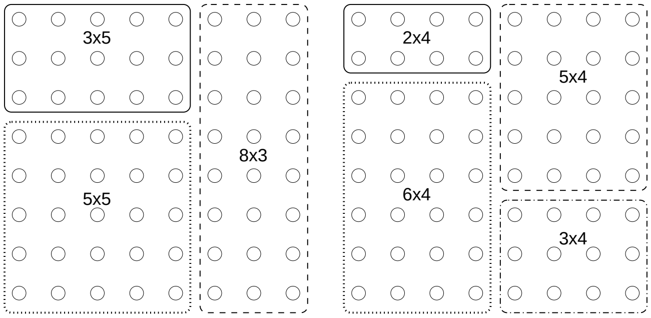
|  | **4x4 Network** |  |  |  |
|---|---|---|---|---|
|  | VALIANT | ROMM | DOR | O1TURN |
| Random | 0.5 | 1 | 1 | 1 |
| Transpose | 0.5 | 0.889 | 0.333 | 0.667 |
| Complement | 0.5 | 0.4 | 0.5 | 0.5 |
| Perf. shuffle | 0.5 | 0.706 | 0.5 | 0.667 |
| Worst-case | 0.5 | 0.364 | 0.333 | 0.5 |
| Avg. case | 0.5 | 0.532 | 0.478 | 0.543 |

Table 1: Normalized Worst case and Average Case Throughput

generation for the various submeshes are independent, it is possible that the saturation throughput of each domain is different. We compute the average saturation throughput of each of the domains with 1000 random permutations. We then compute the average aggregate throughput for the entire 8x8 network by computing a weighted average[4] of the average throughputs of each domain. We present results for two submesh partitions as shown in Figure 5. Table 2 summarizes the normalized average aggregate throughputs for the two local patterns. Note, the average aggregate throughput is normalized to the network capacity of the full 8x8 network for consistency.

The conclusions drawn from local traffic results are similar to those of global traffic: ROMM and O1TURN offer a perceptible improvement over DOR (13%-15%) but it is hard to differentiate between O1TURN and ROMM, especially since the average case throughput estimation technique uses approximation.

[4] The average is weighted by the number of nodes in each submesh.



(a) Pattern A (b) Pattern B

Figure 5: Local Traffic Patterns

| Routing | Pattern | |
|---|---|---|
| | A | B |
| DOR | 0.716 | 0.868 |
| ROMM | 0.816 | 0.984 |
| O1TURN | 0.826 | 0.998 |

Table 2: Local Traffic Patterns on 8x8 Network

### 3.2 General Applicability of Analytical Results

The above analysis limited the network topology to square, 2D mesh networks with independently routed, single flit packets. In this section, we describe O1TURN's properties when the limitations are relaxed to include realistic mesh networks such as rectangular networks, higher dimension networks and variable length packets.

**Rectangular Networks:** All results claimed for O1TURN for square ($k$x$k$) mesh networks hold for rectangular 2D mesh networks (say, a $k_{max}$x$k_{min}$ network or a $k_{min}$x$k_{max}$ network where $k_{max} > k_{min}$) with minor modifications. Analysis proves that O1TURN achieves (a) optimal worst-case throughput for even $k_{max}$ and (b) is within a factor of $(1/k_{max}^2)$ of optimal worst-case throughput if $k_{max}$ is odd. We do not provide a detailed proof of this claim as it is very similar to the proof for square networks.

**Higher dimension mesh networks:** Higher dimension variants of O1TURN do not guarantee near-optimal worst-case throughput. For example, O2TURN for 3D mesh networks that routes packets in one of the 6 possible dimension orders (XYZ, XZY, ZXY, ZYX, YXZ, YZX) does not guarantee the optimal worst-case throughput.

This can be proved by building an adversarial traffic pattern for O2TURN that fails to achieve optimal worst-case throughput. For example, consider a $k$x$k$x$k$ network with even $k$. The network capacity for such a network is also $(4/k)$ using analysis similar to that used in Section 2.2. This implies that the optimal worst-case throughput is $(2/k)$ packets/node/cycle. The channel load corresponding to the worst-case throughput is $(k/2)$.

Consider a traffic pattern where the node $(k/2, k/2, 1)$ is sending packets to $(k/2, k/2, k)$, the node $(k/2, k/2, 2)$ is sending with $(k/2, k/2, k - 1)$ and so on. In general, the node $(k/2, k/2, i)$ sends packets to $(k/2, k/2, k + 1 - i)$ for all $1 \leq i \leq k/2$. The above pattern causes the load on the physical link from $(k/2, k/2, k/2)$ to $(k/2, k/2, k/2 + 1)$ to be $k/2$ since each packet is forced to travel in the minimum rectangle. The channel load corresponding to the optimal worst case throughput in a $k$x$k$x$k$ network is $k/2$. Since the traffic pattern above causes a channel load of $(k/2)$, it implies that any additional channel load results in sub-optimal worst-case throughput.

Now consider any packet that travels from $(k/2, 1, 1)$ to $(1, k/2, k)$. One of the six possible paths considered by O2TURN traverses dimensions in YZX order. Along this path, the intermediate corner vertices are $(k/2, k/2, 1)$ and $(k/2, k/2, k)$. Thus, the packet contributes to additional channel load on the link from $(k/2, k/2, k/2)$ to $(k/2, k/2, k/2 + 1)$, leading to a degradation in throughput.

The above example represents one possible communication pair that can add channel load on the critical channel. There exist many other ways to demonstrate sub-optimality of higher dimension variants of O1TURN.

**Multi-flit and/or Variable Packet Length:** The near-optimal worst-case throughput behavior of O1TURN relies on distributing channel load evenly between the XY and YX routing paths. This may be achieved trivially for networks with fixed packet sizes that are independently routed by randomly picking one of the two choices for each packet. Systems with variable length packets will require additional circuitry to maintain load balance. However, this additional circuitry is completely off the critical path and does not lead to any additional delay complexity. We explain this issue in greater detail in Section 4.3 in the context of overall delay analysis of the O1TURN router.

**Summary**: In this section, we described the O1TURN routing algorithm and proved that it achieves near-optimal worst-case throughput for 2D mesh networks. It is optimal for $k$x$k$ networks where $k$ is even and within a $(1/k^2)$ term when $k$ is odd. O1TURN also offers similar average performance as ROMM under both global and local network traffic. In the next section, we examine a router implementation for O1TURN.

# 4 Router Implementation

We consider a virtual channel router as our base router model. Figure 6 illustrates the organization of our base router model with multiple virtual channels [4, 5] per physical channel. In this section, we first provide an overview of the operation of our base router (Section 4.1). A description of a router for O1TURN follows in Section 4.2. Finally, we examine the router delay for our implementation in Section 4.3.

## 4.1 Base Router Model

In previous sections, we had assumed an atomic single cycle router and single-flit packets to simplify discussion of the throughput analysis method. We discard those simplifying assumptions in favor of a more realistic model where each packet is composed of multiple “flits” (flow control units) and the router is pipelined.

Each packet arrives on a physical input port and is delivered to the appropriate virtual channel buffer (input buffer associated with a virtual channel) by examining the *virtual channel identifier (VCID)* included with each flit’s control information. (See Figure 6). It then progresses through various stages in the router before it is delivered to a neighboring router. The first pipeline stage—the *routing* stage—determines the packet’s possible output physical channels. Next, the packet is in the *VC-allocation* state where the allocator examines all input packets and their destinations and assigns free output virtual channels to the packets, if available. Third, the packet competes for a switch port in the switch arbitration[5] stage. Finally, on switch grant, the packet is delivered

[5] Also referred to as switch allocation

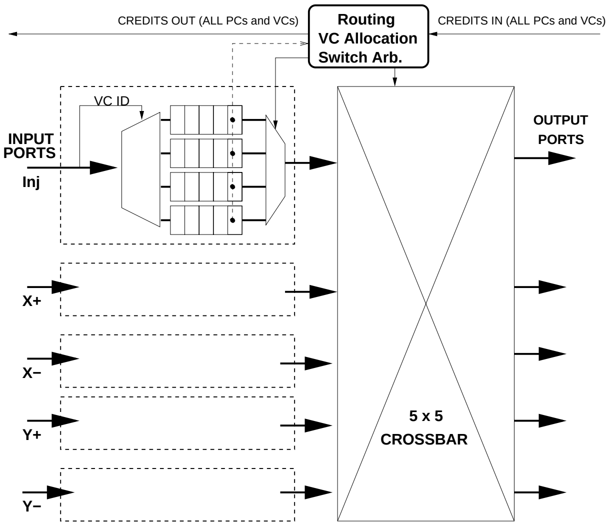


Figure 6: Base Router Model

across the crossbar and the physical channel to the neighboring node by the virtual channel controllers. Our router pipeline model assumes that there's a one cycle delay after a packet is delivered to the neighboring node before it's input buffer is marked free to allow for credit propagation. We do not present the implementation of the VC allocator and switch arbiter in this paper but we assume an implementation as described by Peh *et.al.* [15].

## 4.2 The O1TURN Router

With the base router model described above as a starting point, we describe the modifications necessary to implement O1TURN routing. Since one virtual channel is sufficient to achieve deadlock-free DOR routing in a mesh network, O1TURN can be implemented with two virtual channels (VCs) per physical channel(PC) by using one VC for XY routing and another for YX routing. In general, half of the virtual channels can be used for XY routing "layer" and the other half for YX routing "layer".

Once a packet is injected into the network, the packet is constrained to remain within the virtual channels of the layer it was injected into. Figure 7 illustrates our router implementation for O1TURN. The routing and VC allocation blocks are duplicated with one of them responsible for routing and VC allocation for the packets in the XY-layer and another responsible for the same functions in the YX layer. Routing and VC allocation for the two layers are completely independent of each other. The shaded buffers constitute the virtual channels associated with the Y-X routing layer.

The VC allocation logic, which matches free VCs on a neighboring node to requests from packets in the input queues in the local node, is simplified because each layer's VC allocation logic has to deal with half as many contenders and half as many VCs to allocate. However, the switch arbitration is still global over all virtual channels since packets from all layers have to multiplexed over the same crossbar ports.

The partitioned organization as shown in Figure 7 reduce the number of virtual channels per layer by a factor of two when compared to a non-partitioned, monolithic DOR router. This can potentially be a problem because available VC prevent head-of-line (HOL) blocking. One conservative method to equalize this potential performance problem across the two classes of routers is to compare the delay of a DOR router with $n$ VCs with an O1TURN router with $2n$ VCs. This equalization is unfavorable to O1TURN when modeling delays since increasing the number of virtual channels, in general, increases delays. The next section will show that O1TURN suffers from no delay penalty in spite of this conservative handicap.

Note, this equalization of VCs per routing layer is conservative in the context of delay modeling. However, this would introduce bias favorable to O1TURN when evaluating performance by simulation. As such, we do *not* carry over this assumption to our simulation methodology as explained in Section 5.

### 4.3 Delay Analysis

We adopt existing delay models for pipelined routers described by Peh and Dally [15] to estimate the delay through our pipeline stages in terms of FO4[6] delays. These models are based on the logical effort method [25] and have been validated using Synopsis timing simulations in the original paper [15].

Table 3 tabulates the delays of the VC allocation and switch arbitration stages in terms of the FO4 units

[6] The FO4 unit is a technology-independent metric to measure circuit delay. It refers to the delay through an inverter with a fan-out of four.

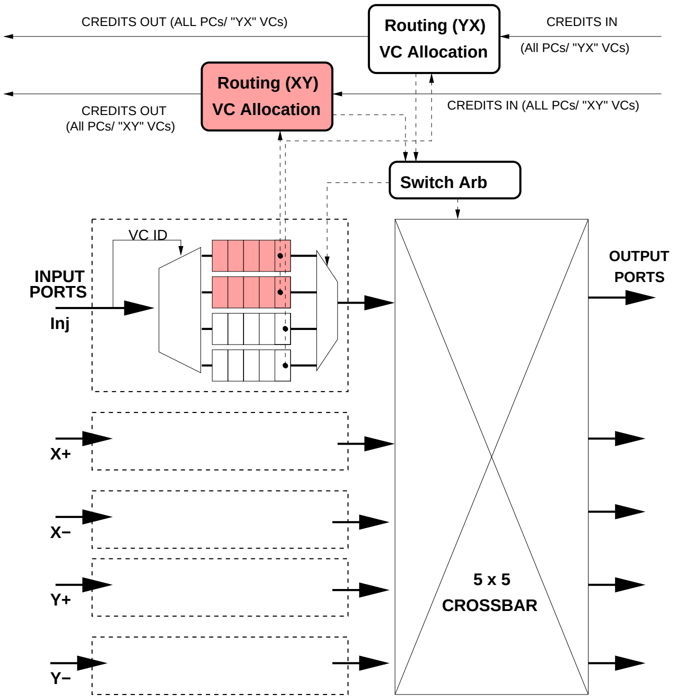


Figure 7: O1TURN Router Implementation

achieved by the base DOR router and the O1TURN router as obtained from the delay model. The first column lists virtual channels (VCs) per physical channel (PC) per routing layer rather than the aggregate VC/PC. Because O1TURN has two routing layers (as opposed to one for DOR), the table effectively compares the delay of a 4VC/PC (8VC/PC) O1TURN router with that of a 2VC/PC (4VC/PC) DOR router.

Comparing the delay across columns on a single row of Table 3, the following observations can be made:

- The VC-allocation delay does not change. This is not surprising because of the partitioned VC allocation logic in the O1TURN router. The delay depends on the number of contenders/outputs being arbitrated and they are equal under our equalization rule.

- The switch arbitration delay sees a slight increase in O1TURN as compared to DOR. This is expected because switch arbitration has to deal with twice as many contenders in the O1TURN router as the DOR router.

- In spite of the increase in switch arbitration delay, the VC-allocation stage remains the critical stage in both

| VCs/PC/ | DOR | | O1TURN | |
|---|---|---|---|---|
| layer | VC Alloc | Sw. Arb | VC Alloc | Sw. Arb |
| 2 | 14 | 11 | 14 | 14 |
| 4 | 17 | 14 | 17 | 16 |

Table 3: Router Pipeline Delays (FO4)

configurations.

The observation imply that any clock cycle that accommodates the VC allocation will also accommodate the increased switch arbitration delay. Consequently, the penalty of the additional virtual channels is completely hidden in pipelined routers. Thus, the delay through O1TURN is no worse than a DOR router.

For non-pipelined routers, or for routers where VC-allocation is not the clock critical stage, O1TURN router may incur a marginal delay penalty. (Note, this is under the adverse equalization criteria where O1TURN has twice as many VCs as the base router.) But latency hiding techniques can effectively hide the increase in switch arbitration delay in non-pipelined routers as well [12].

**Load Balancing with variable length packets:**

Recall, the worst-case throughput fundamentally requires channel load to be equally distributed between the two routing layers. This is challenging in systems with variable packet sizes (or packet classes that cannot be routed independently to ensure in-order delivery guarantees). Load balance can be achieved in such systems by tracking the number of flits injected in a signed counter that is incremented when flits are injected into the XY plane and decremented when the flits are injected into the YX plane. The choice of injection layer for every new packet depends on the sign-bit of the counter which indicates the excess/deficit of flits injected into one of the layers. This can be computed in advance and kept available before the next packet's injection. As such, this mechanism is completely off the critical path and does not affect our delay analysis.

# 5 Evaluation Methodology

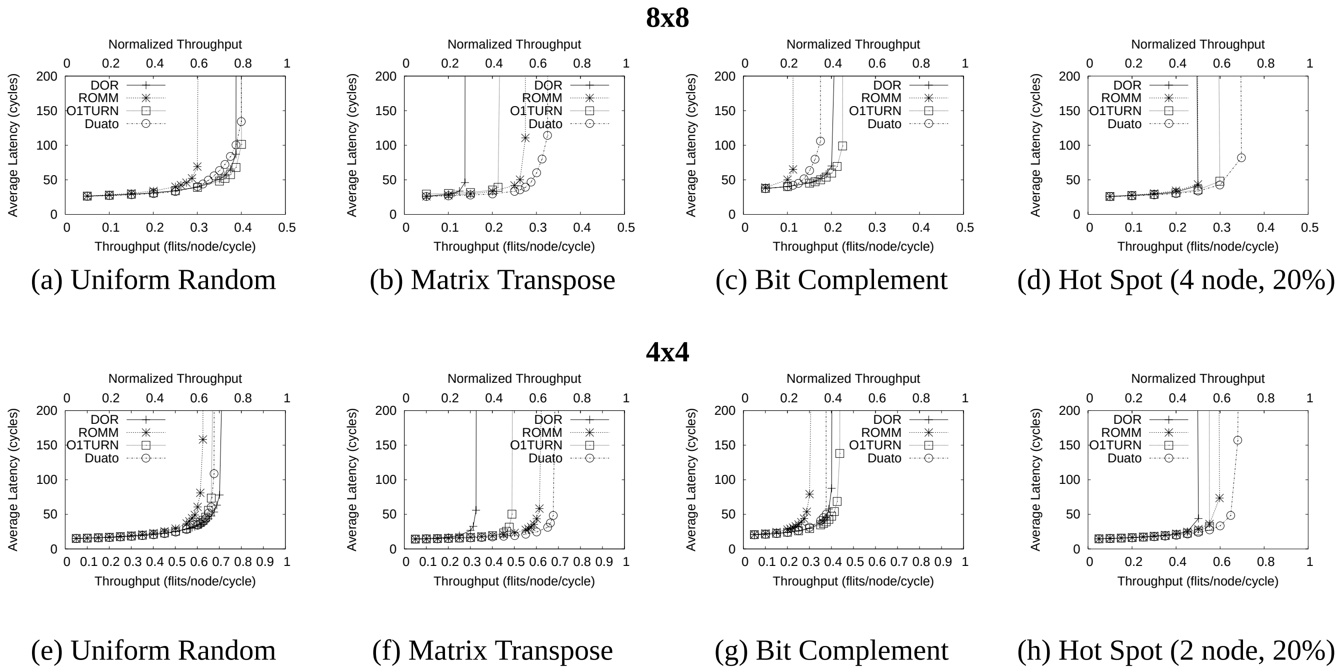
The worst-case throughput results are an upper bound on performance. It is necessary to measure the actual delivered throughput under realistic conditions with the router model we have assumed. We use simulation to evaluate the O1TURN router. We use a modified version of the PoPnet [20] network simulator. PoPnet simulator models a four-stage router pipeline with the following four stages: routing, VC allocation, switch arbitration/allocation and traversal through the cross-bar and physical channel. We assume that the logic delay of the pipeline stages and the physical traversal to the neighboring node can be accommodated within one clock cycle each. The network was simulated for 500,000 cycles. The reported latency includes queuing time before a packet is injected into the network and is measured till the tail flit is drained at the destination node.

We present results for two network sizes: 4x4 and 8x8. The channels are full-duplex bidirectional links. The base DOR router and O1TURN router is modeled after the architecture illustrated in Figure 6 and Figure 7, respectively. We assume 8 virtual channels (VCs) per physical channel (PC)[7] and that each input buffer can hold 5 flits. We use 5-flit packets. Finally, we also include an adaptive routing algorithm based on deadlock avoidance [7], DUATO, in the comparison. We chose the DUATO routing protocol as our experiments indicate that it achieves better perfomance compared to PFNF. The packet size and buffer size are the same network parameters identified by Wang *et.al.* [33] as being representative approximations of the onchip networks of MIT RAW and TRIPS [16].

Note, we use the same number of VCs/PC for both O1TURN and DOR. The assumption made in Section 4.3 on equalizing VCs/PC per layer introduces bias in favor of O1TURN. As such, we consider the same number of total VCs/PC in this section.

We simulate a steady offered load at various injection rates. We consider three different traffic permutations, *uniform random, complement and matrix transpose*. These communication patterns differ in the way a

[7] We deliberately chose a high number of virtual channels per physical channel, not to be generous to O1TURN- but to eliminate HOL blocking in ROMM. ROMM uses additional virtual channels for handling deadlock-free two-phase routing. Thus its performance suffers adversely when there are fewer VCs/PC.



(a) Uniform Random (b) Matrix Transpose (c) Bit Complement (d) Hot Spot (4 node, 20%)

(e) Uniform Random (f) Matrix Transpose (g) Bit Complement (h) Hot Spot (2 node, 20%)

Figure 8: Router Performance

destination node is chosen for a given source node with bit co-ordinates $(a_{n-1}, a_{n-2},\ \ldots, a_1, a_0)$. The bit co-ordinates for the destination nodes are $\overline{(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)}$ for*complement* and $(a_{(n/2)-1}$ , $a_{(n/2)-2}$ , . . . , $a_0$, $a_{n-1}, a_{n-2} \ldots, a_{(n/2)0}$ for *matrix-transpose*.

We also consider one “hot-spot” traffic pattern wherein 20% of overall traffic is sent to 2 (in case of the 4x4 network) or 4 (in case of the 8x8 network) randomly-chosen nodes. constraints, We also present a larger set of simulation results in which we vary (a) the number of hotspots, (b) placement of hot-spots in the network and (c) the fraction of traffic that is sent to the hot-spots in Appendix B.

# 6 Results

The primary conclusion from our simulations is that the performance of O1TURN is closer to the best of DOR and ROMM. In cases where ROMM outperforms DOR, O1TURN is close to or better than ROMM. The same is true when DOR achieves better throughput than ROMM. We also observe that DUATO does achieve higher throughput in the case of hot-spot traffic, but this comes at the cost of increased delay complexity (see Section 6.1).

The simulation results are presented in Figure 8 Figs 8(a)–(d) correspond to an 8x8 network size and Figs 8(e)–(h) are for a 4x4 network. There is one graph for each of the four (three permutation and one hot-spot) traffic patterns. Each graph has four curves – one each for DOR($+$), ROMM($*$), O1TURN($\Box$) and DUATO($\circ$). Each graph has delivered throughput on the x-axis at two different scales: (a) the x-axis below specifies the absolute value (flits/node/cycle) of throughput and (b) the x-axis above normalizes delivered throughput to network capacity. The graphs plot average packet latency in cycles on the Y-axis. For any given curve, starting from low loads, as the load increases, the delivered throughput increases without much increase in latency. No latency increase is expected before saturation because network packets do not experience contention. However, when load approaches saturation throughput, we observe a sudden and sharp increase in latency. The curve that saturates at the highest load represents the better router.

We observe that, with the exception of the 4x4 network with uniform random traffic (Figure 8(e)) ROMM or O1TURN consistently outperform DOR. The results for uniform random traffic on the 8x8 network (Figure 8(e)) shows that DOR performs the best under this traffic pattern. This is not surprising because random traffic is an “easy” communication pattern. As such, any difference we see in the graphs is because of variation in HOL-blocking for the different routing algorithms. ROMM suffers more than DOR or O1TURN because it uses additional virtual channels for deadlock-free routing which reduces the number of VCs available for HOL-blocking.

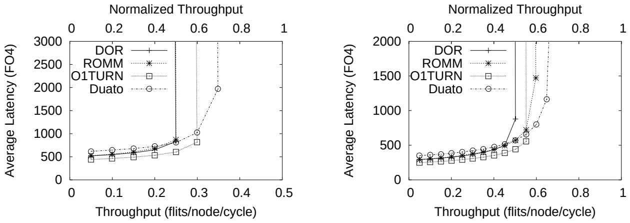
Note, in the case of bit-complement traffic (Figure 8(c) and Figure 8(g)), O1TURN saturates at approximately the same (or better) load as DOR and DUATO. This is not surprising because every packet crosses the network bisection in the bit-complement traffic pattern. This imposes a hard-limit of 50% of network capacity as the maximum possible throughput for *any* routing algorithm. As such, O1TURN is unable to improve upon DOR. ROMM is worse than DOR as well and this was expected as per the analytical results. (See Table 1.)

On the hotspot pattern (Figure 8(d) and Figure 8(h)), DUATO does outperform the oblivious routing algorithms. However, the simulations results presented above were stated in terms of cycles. This introduces a bias in favor of DUATO because the cost of additional delay complexity is not reflected in the graphs. In the next section, we incorporate the delay complexity and re-examine the latency/bandwidth tradeoff.

### 6.1 Delay penalty of adaptive routing

We obtain the delays for the various pipeline stages of the router for the 8VC/PC configuration using the Peh-and-Dally model [15]. We assume that the clock cycle is purely determined by the critical router pipeline stage. The original router delay model [15] assumes a fixed 20 FO4 clock cycle as an externally imposed constraint. We do not assume any external constraints.) The clock cycle times for DOR, ROMM, O1TURN and DUATO are 20, 20, 17 and 24 FO4 delays respectively. Note, the increased delay complexity of adaptive routing is *not* because of the routing stage. On the contrary, the routing stage is not the critical stage because it only involves comparison of mesh co-ordinates as opposed to the multi-stage arbiter structure for the VC Allocation and arbitration stages in the pipeline. It is the fact that adaptive routing algorithms offer more than one possible output physical channel to the VC-allocation stage (unlike oblivious routing algorithms that offer only one possible output physical channel) that causes the VC allocation stage of adaptive routers to suffer from additional delay. Using the logical effort model, we show elsewhere that even with a naive implementation of the routing stage, the clock-determining stage remains the VC allocation stage [19].

Figure 9 replots the latencies of the various routing algorithms in absolute FO4 units for the hot-spot pattern for the 4x4 network configuration. (Similar graphs for all other traffic patterns are included in Appendix B .) Figure 9 clearly illustrates the delay penalty of adaptive routing. We observe that the curve for DUATO is shifted upwards. More importantly, DUATO suffers from the latency penalty even at low loads. A well-designed network must operate below the saturation load in the common case and thus, the increased latency penalty due to adaptive routing is incurred even in the common case.



(a) 8x8 (b) 4x4

Figure 9: Absolute latency(FO4)/throughput for Hot Spot Traffic

# 7 Related Work

A significant part of the related work is discussed in the background section. As such, we restrict discussion in this section to related work not described elsewhere in this paper to minimize redundancy.

Peh *et.al.* [15] describe a speculative pipelined router to overlap the delays of VC allocation and switch arbitration to reduce latency. Our technique reduces the complexity of the critical stage: VC allocation which may further improve router latency. Mullins *et.al.* describe an architecture to remove VC-allocation and switch arbitration from the critical path through the router [12]. Towles *et.al.* cast the problem of oblivious router design as a linear program [30]. The insights from their linear program framework characterizes the optimal latency/throughput trade-off for both worst-case and average case throughput.

Upadhyay *et.al.* combine positive-first and negative-first flavors of *"Turn model"* [8] based routing to create a load balanced, adaptive algorithm called PFNF routing [31]. This is similar to the use of two heterogeneous routing layers in O1TURN. PFNF is an adaptive routing algorithm and thus incurs a latency penalty due to the complexity of adaptive routing. Further, because O1TURN is provably near-optimal, the worst-case performance of PFNF algorithm cannot be better by any significant margin.

Jesshope *et.al.* use per-quadrant virtual networks that are individually deadlock-free in their "mad postman" routing mechanism [9]. While the use of virtual networks is similar to O1TURN's architecture, this does not eliminate the delay penalty because adaptive routing may offer more than one output physical channel to the VC allocation stage. We do not compare the delay of O1TURN router to the implementation described in the mad-postman paper [9] as it includes orthogonal issues and implementation artifacts such as (a) speculative routing to minimize switching delay over bit-serial physical links and (b) a 9-cycle delay between adaptively trying the second output physical channel.

Choi and Pinkston describe various partitioned crossbar architectures for adaptive, deadlock-recovery based routers by exploiting "routing locality"– the property that a packet tends to traverse the network in the same virtual/physical channel [2]. However, their technique assumes an aggressive PVxPV (where P is the number of physical channels and V is the number of virtual channels per physical channel) crossbar rather than the simpler PxP crossbar we use in our base router architecture. As such, their technique is not directly applicable to our base router model.

## 8 Conclusion

Mesh networks constitute an important class of interconnection networks. They are popular for various domains such as switch fabrics, multiprocessor interconnection networks and on-chip networks. Good worst-case and average-case throughput, minimal number of network hops and low complexity router implementation are all desirable goals in the design of mesh networks. The key contribution of this paper is the design of a routing algorithm—O1TURN—that addresses the challenge of satisfying all the above design goals for two dimensional mesh networks.

With O1TURN routing, packets are routed using one of at-most two minimal, dimension-ordered paths by randomly choosing the first traversal dimension. O1TURN achieves near-optimal worst-case throughput and good average case throughput. When the network radix (i.e., $k$ in a $k$x$k$ network) is even, O1TURN is provably optimal and no routing algorithm, oblivious or adaptive, minimal or non-minimal, can achieve better worst-case throughput. The worst-case throughput is within $(1/k^2)$ of the optimal worst-case when $k$ is odd. Though it is less-flexible than ROMM in terms of the number of possible routes a packet may traverse, O1TURN achieves comparable average case throughput as ROMM and much better worst-case throughput. On the other hand, VALIANT, which achieves optimal worst-case throughput compares poorly with O1TURN on other metrics such as latency and average case behavior. O1TURN lends itself naturally to a simple, partitioned implementation. Delay analysis of the proposed implementation using a pre-existing delay model [15] demonstrates that the delay through clock-critical circuitry of the partitioned router is comparable to, if not better than, an equivalent dimension-ordered router. Finally, simulations confirm that the saturation throughput of O1TURN is as expected from the analysis (modulo the reduction from the theoretical limit due to realistic implementation).

In summary, O1TURN is an attractive design point for two dimensional mesh networks offering near-optimal worst-case throughput, good average throughput, low latency (due to minimal routing) and low-complexity router implementation.

# 9 Acknowledgements

We would like to thank the anonymous referees for their feedback and suggestions to improve this paper. We wish to thank T.N. Vijaykumar, Michael Powell, Ethan Schuchman, Zeshan Chishti, Jahangir Hasan and Ankit Jalote for the useful discussions on this work. We would also like to thank Li Shang for help with the Popnet simulator. This work is supported in part by Purdue Research Foundation XR Grant No. 6904010 and Purdue University.

# References

[1] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B.-H. Lim, K. Mackenzie, and D. Yeung. The mit alewife machine: architecture and performance. In *25 years of the international symposia on Computer architecture (selected papers)*, pages 509–520. ACM Press, 1998.

[2] Y. Choi and T. M. Pinkston. Evaluation of crossbar architectures for deadlock recovery routers. *J. Parallel Distrib. Comput.*, 61(1):49–78, 2001.

[3] D. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, 1999.

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

[5] W. J. Dally and C. L. Seitz. The TORUS routing chip. *Journal of Distributed Computing*, 1(3):187–196, October 1986.

[6] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the Design Automation Conference*, pages 684–689, Las Vegas, NV, June 2001.

[7] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

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

[9] C. R. Jesshope, P. R. Miller, and J. T. Yantchev. High performance communications in processor networks. In *ISCA '89: Proceedings of the 16th annual international symposium on Computer architecture*, pages 150–157. ACM Press, 1989.

[10] R. K. Koeninger, M. Furtney, and M. Walker. A shared memory mpp from cray research. *Digital Technical Journal*, 6(2):8–21, 1994.

[11] A. K.V. and T. Pinkston. An Efficient, Fully Adaptive Deadlock Recovery Scheme : Disha. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 201–210, June 1995.

[12] R. Mullins, A. West, and S. Moore. Low-Latency Virtual Channel Routers for On-Chip Networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (to appear)*, Jun 2004.

[13] T. Nesson and S. L. Johnsson. Romm routing on mesh and torus networks. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pages 275–287. ACM Press, 1995.

[14] P. R. Nuth and W. J. Dally. The j-machine network. In *Proceedings of the 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors*, pages 420–423. IEEE Computer Society, 1992.

[15] L. S. Peh and W. J. Dally. A Delay Model and Speculative Architecture For Pipelined Routers. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA-7)*, pages 255–266, January 2001.

[16] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore. Exploiting ilp, tlp, and dlp with the polymorphous trips architecture. In *Proceedings of the 30th annual international symposium on Computer architecture*, pages 422–433. ACM Press, 2003.

[17] S. L. Scott and G. Thorson. The cray t3e network: Adaptive routing in a high performance 3d torus. In *HOT Interconnects IV*, August 1996.

[18] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *ISCA '05: Proceedings of the 32nd annual international symposium on Computer architecture*, pages 432–443, 2005.

[19] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. Technical Report TR-ECE 05-03, Purdue University, 2005.

[20] L. Shang, L. S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proceedings of the 9th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 79–90, Feb 2003.

[21] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. Goal: a load-balanced adaptive routing algorithm for torus networks. In *Proceedings of the 30th annual international symposium on Computer architecture*, pages 194–205. ACM Press, 2003.

[22] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. Adaptive Channel Queue Routing on k-ary n-cubes. In *Proceedings of the Sixteenth Symposium on Parallel Algorithms and Architectures*, June 2004.

[23] A. Singh, W. J. Dally, B. Towles, and A. K. Gupta. Locality-preserving randomized oblivious routing on torus networks. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 9–13. ACM Press, 2002.

[24] H. Sullivan and T. R. Bashkow. A large scale, homogeneous, fully distributed parallel machine, i. In *Proceedings of the 4th annual symposium on Computer architecture*, pages 105–117. ACM Press, 1977.

[25] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kauffman Publishers, 1999.

[26] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin. Wavescalar. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, page 291. IEEE Computer Society, 2003.

[27] M. Taylor, W. Lee, S. Amarainghe, and A. Agarwal. Scalar Operand Networks: On-chip interconnect for ILP in Partitioned Architectures. In *Proceedings of the International Symposium on High Performance Computer Architecture*, pages 341–353, 2003.

[28] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The raw

microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[29] B. Towles and W. J. Dally. Worst-case Traffic for Oblivious Routing Functions. *Computer Architecture Letters*, 1, February 2002.

[30] B. Towles, W. J. Dally, and S. Boyd. Throughput-centric routing algorithm design. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 200–209. ACM Press, 2003.

[31] J. Upadhyay, V. Varavithya, and P. Mohapatra. A Traffic Balanced Adaptive wormhole routing scheme for Two-Dimensional Meshes. *IEEE Transactions on Computers*, pages 190–197, May 1997.

[32] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 263–277. ACM Press, 1981.

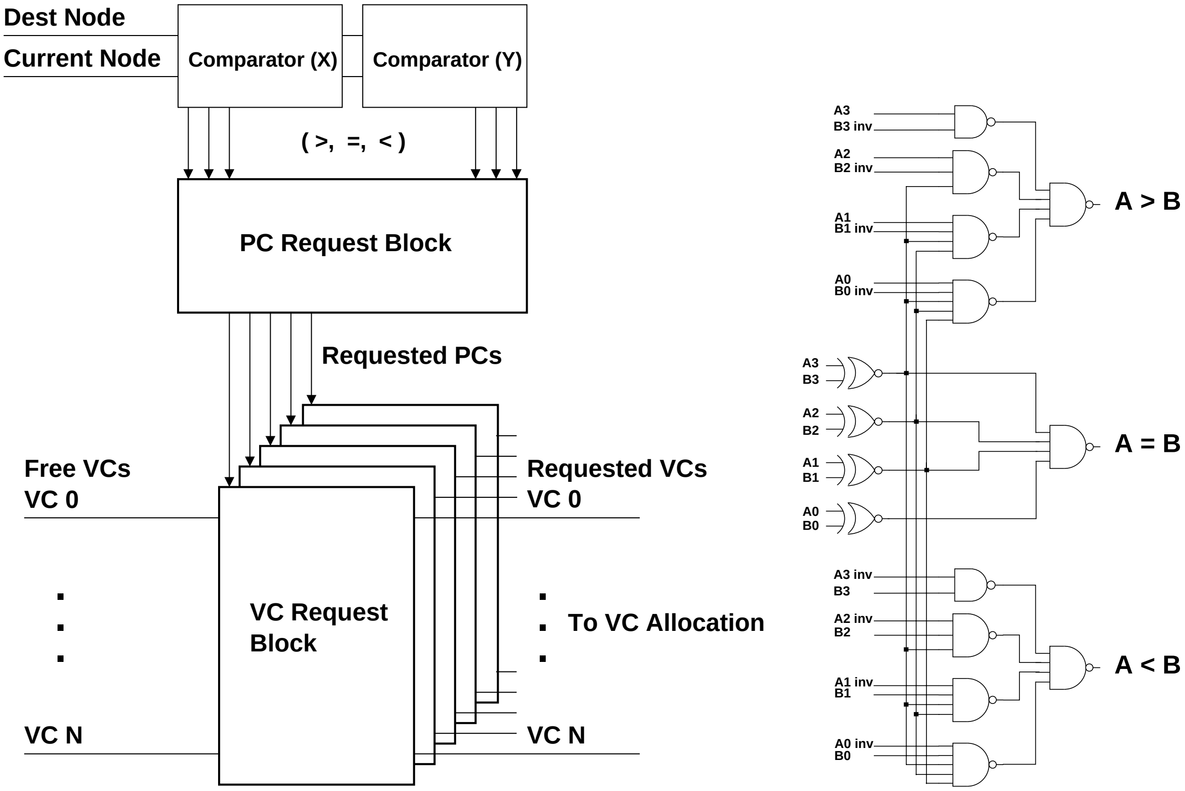
[33] H. Wang, L.-S. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, page 105. IEEE Computer Society, 2003.

# A Delay of Routing Logic

The routing logic for regular networks such as 2D mesh networks is typically simple, comparator-based logic to compare the packet's destination co-ordinates with the current node's co-ordinates. Figure 10 presents one implementation of the routing logic assuming 4-bit node coordinates in each dimension. Four-bit dimension co-ordinates allow for networks upto 16x16 in size.

Consider the following example to illustrate the operation of the routing logic. A packet currently at node $(x_1, y_1)$ with the final destination of $(x_d, y_d)$ does not have to traverse the X-dimension if $x_1$ is equal to $x_d$. On the other hand, if $x_d$ is greater than $x_1$, it has to proceed towards the positive X direction. Otherwise, (i.e., if $x_d < x_1$) it must proceed along the negative X direction. For DOR routing, the difference in Y co-ordinates ($y_d - y_1$) is not relevant for routing till the difference in X co-ordinates goes to zero. (The routing logic of adaptive routing is further simplified by the independence of the routing logic along the X and Y dimensions.) Also, the routing logic has to assert requests for the ejection channels when both X and Y coordinates match at a router. This additional logic that determines specific physical channels to request afte examining the comparator output is easily implementable using 2 levels of 2-input logic gates. This logic block is abstractly shown as the *PC Request* block. Finally, the *VC request* block generates the final list of VC requests (as inputs to the VC allocation stage) for the packet by performing a logical AND of the requested physical channels with the free virtual channels. (The physical channels requested may be latched since they do not have to be recomputed every cycle.)

Analysis using the logical effort delay model [25] demonstrates that the delay through the routing logic does not exceed 12 FO4 even in designs with 16 VCs. Recall that the VC Allocation stage incurs a minimum of 14 FO4 delay (See Section 4.). As such we conclude that the routing stage is not clock-critical.



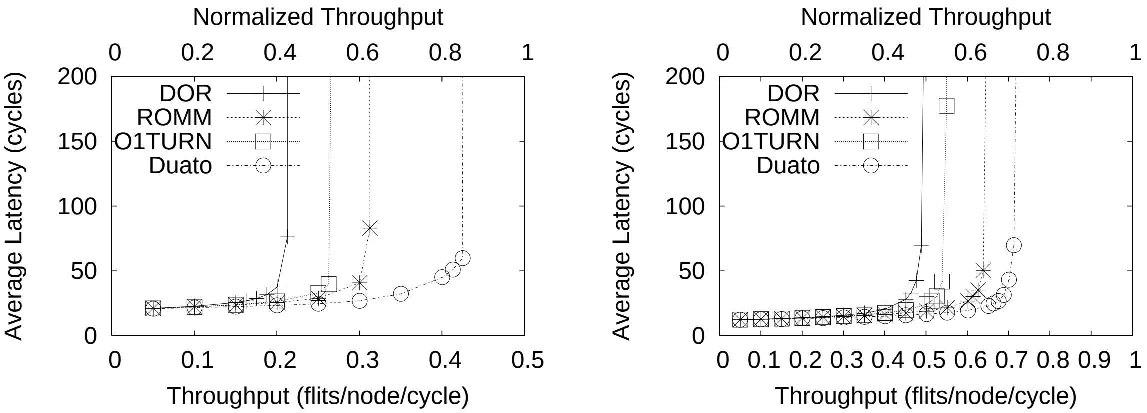
(a) Block Diagram

(b) Comparator network

Figure 10: Routing Logic

## B Other results

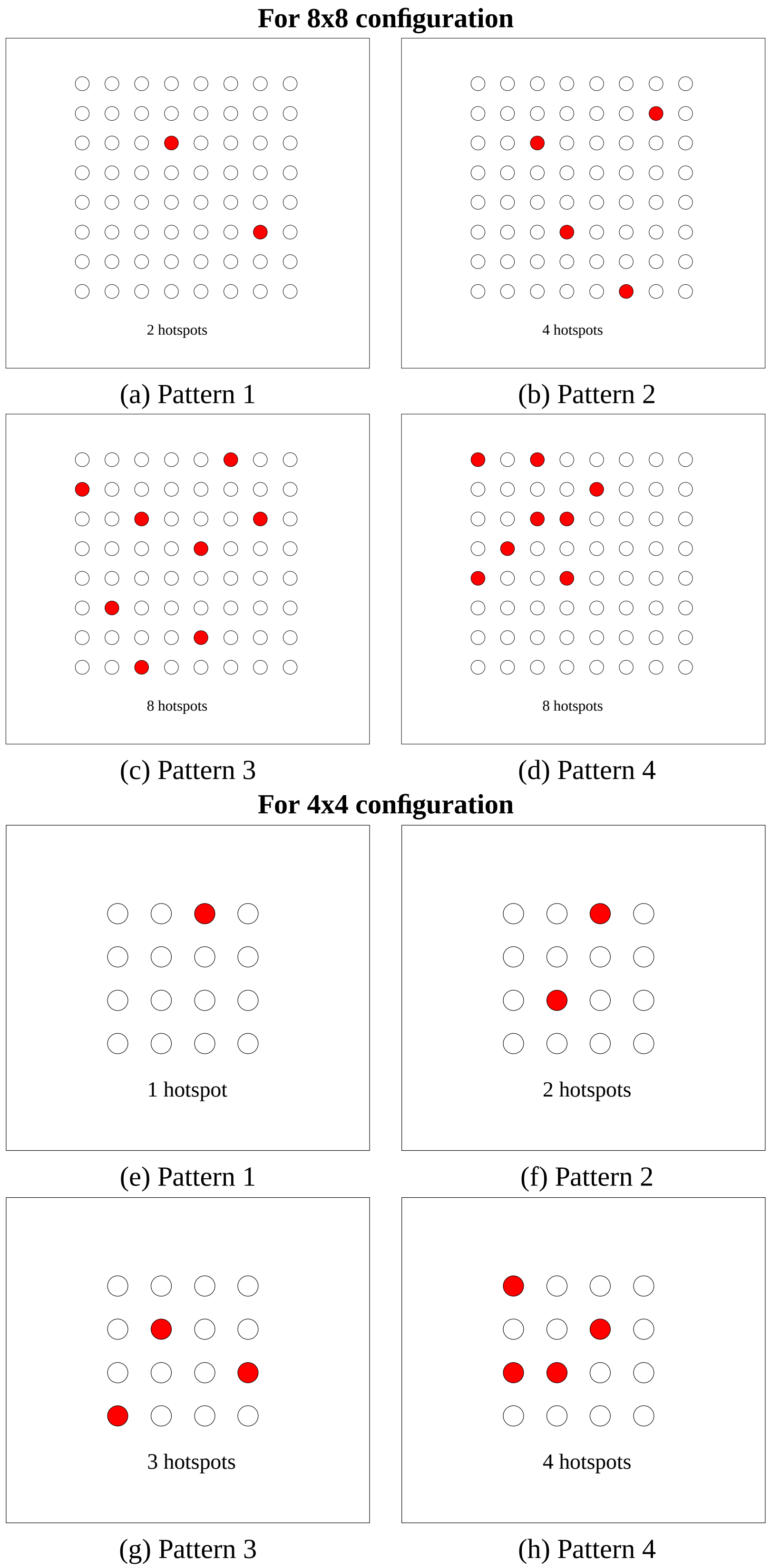
This section contains results for traffic patterns not included in Section 6. Figure 11 plots the performance of O1TURN and other routers for the *perfect-shuffle* communication pattern. We also present exhaustive results for various hotspot placement patterns (as shown in Figure 12). For each of the hotspot placement pattern, we also vary the fraction of traffic that is directed towards the hotspots by changing this parameter to 10% (Figure 13), 20% (Figure 14), 30% (Figure 15) and 40% (Figure 16).



(a) 8x8 network

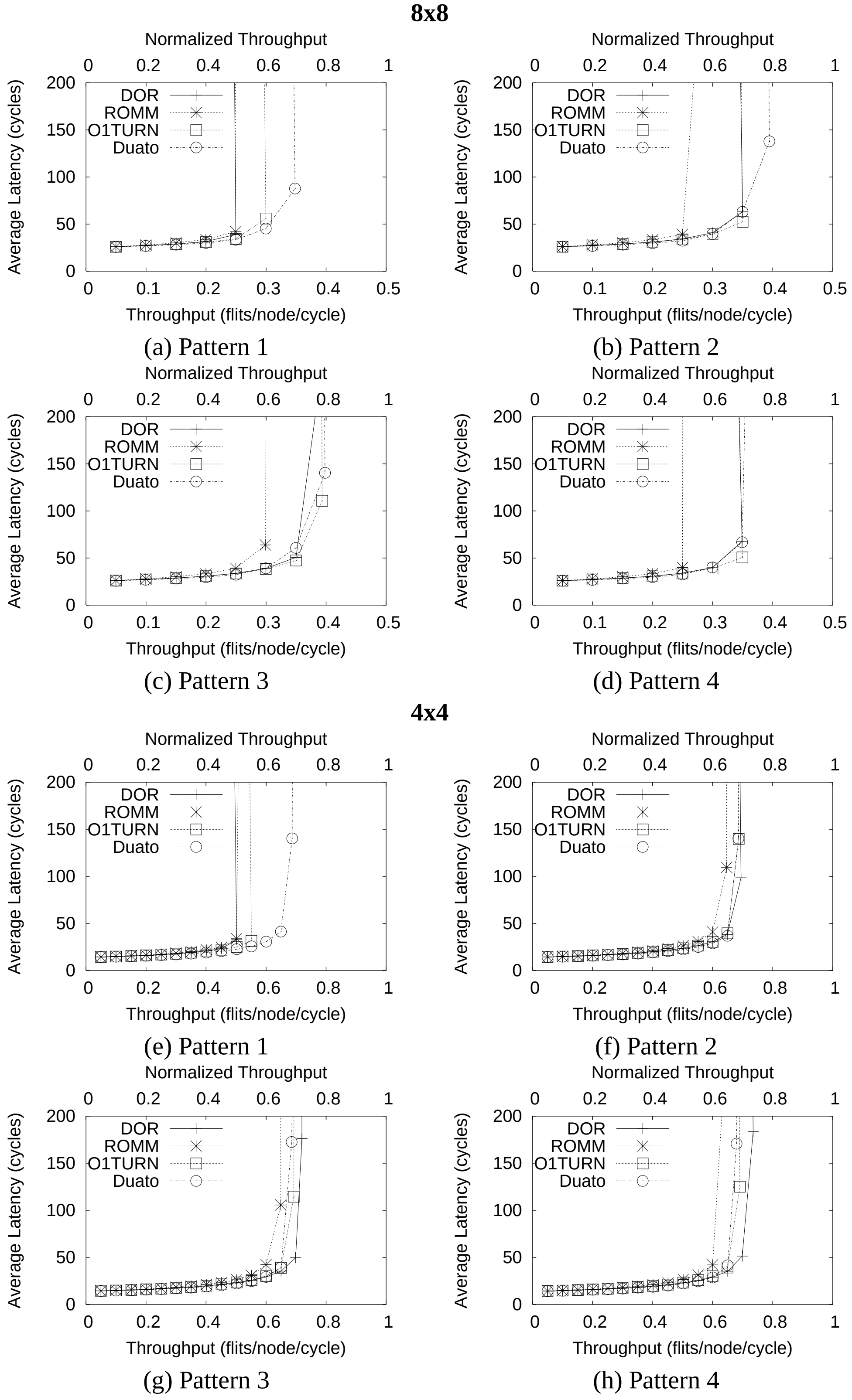
(b) 4x4 network

Figure 11: Performance with Perfect Shuffle Traffic Pattern



(a) Pattern 1 (b) Pattern 2 (c) Pattern 3 (d) Pattern 4 (e) Pattern 1 (f) Pattern 2 (g) Pattern 3 (h) Pattern 4

Figure 12: Hot Spot Placement Patterns



(a) Pattern 1

(b) Pattern 2

(c) Pattern 3

(d) Pattern 4

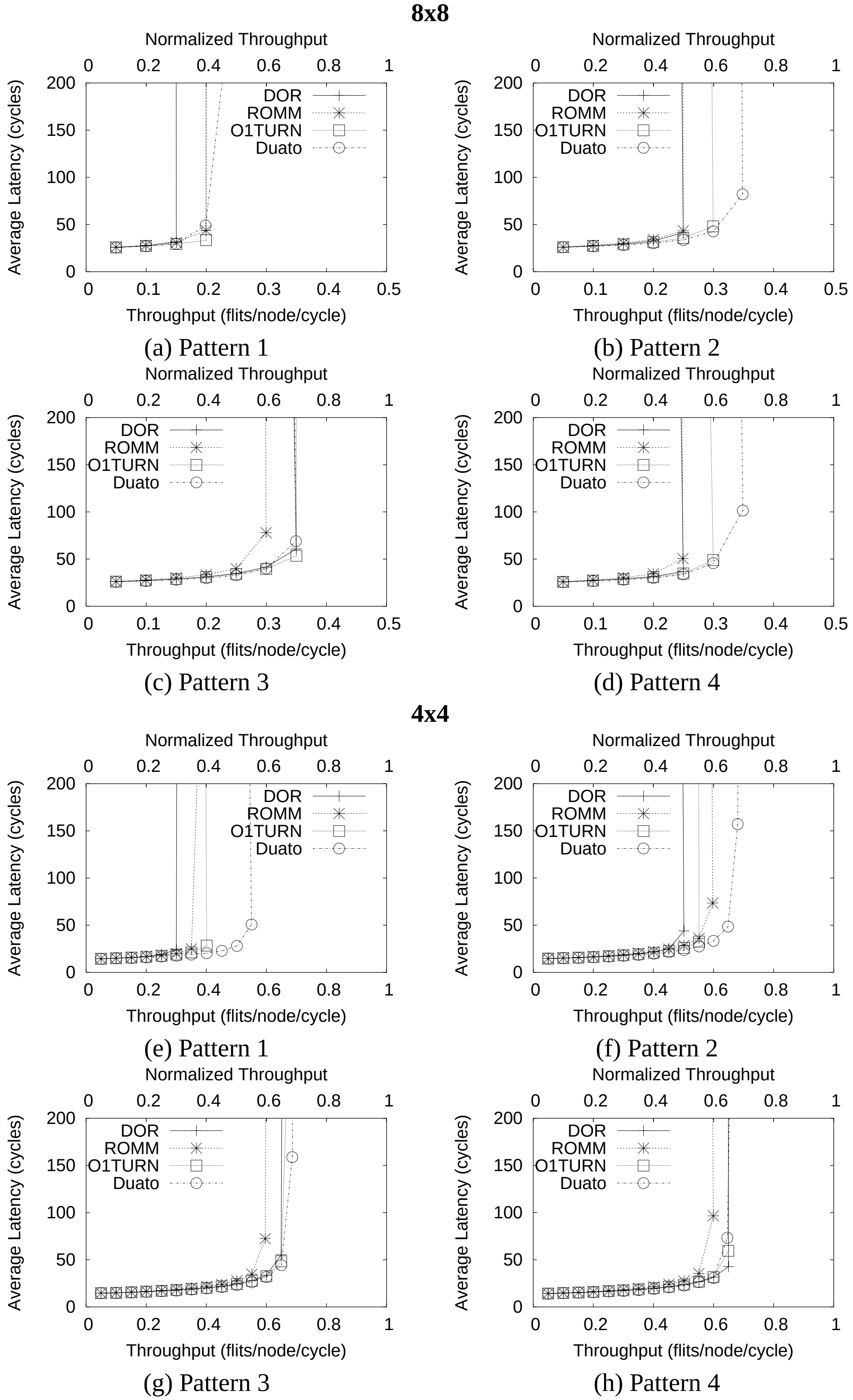
(e) Pattern 1

(f) Pattern 2

(g) Pattern 3

(h) Pattern 4

Figure 13: Hot Spot Performance with (10% HS traffic)



(a) Pattern 1

(b) Pattern 2

(c) Pattern 3

(d) Pattern 4

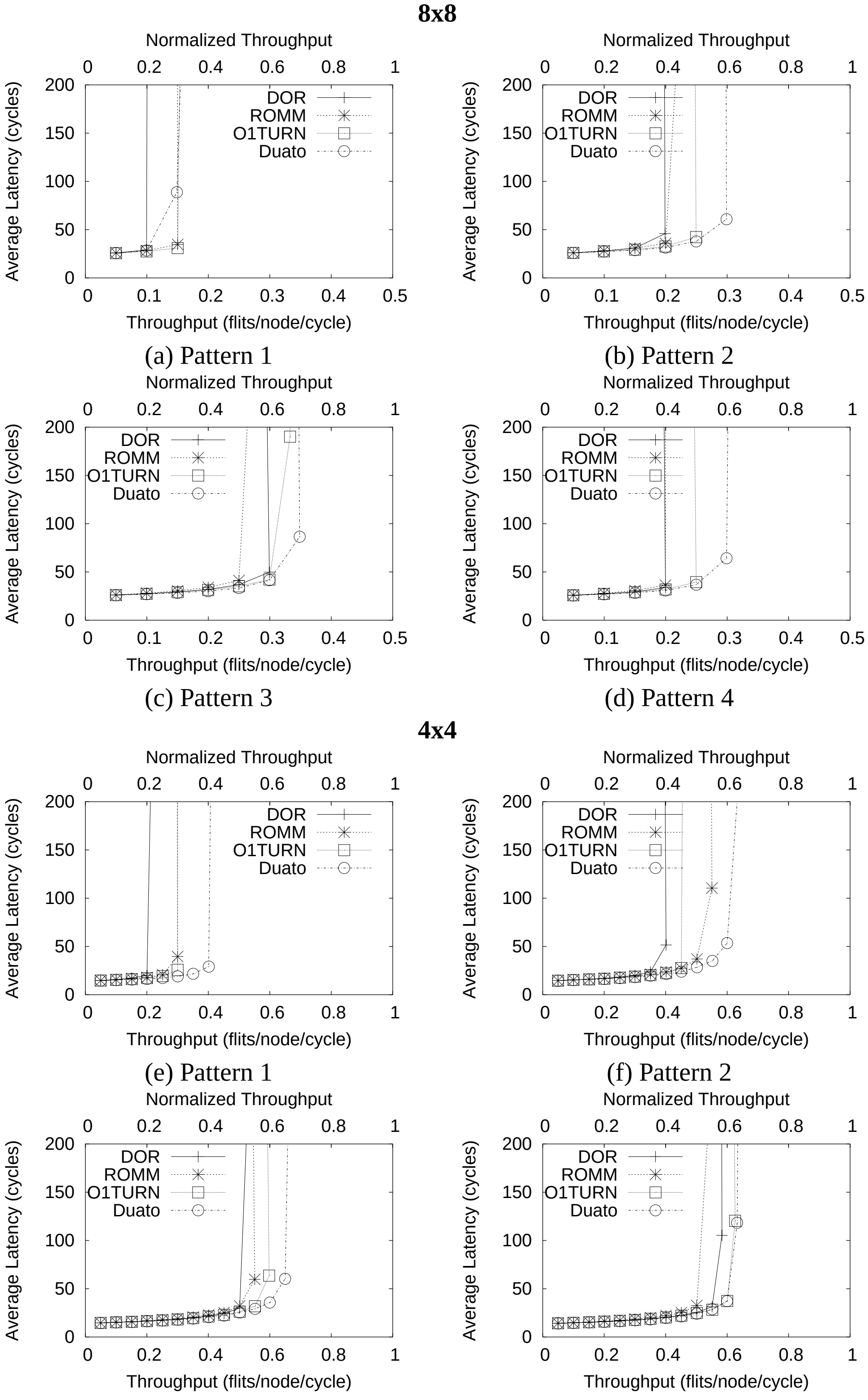
(e) Pattern 1

(f) Pattern 2

(g) Pattern 3

(h) Pattern 4

Figure 14: Hot Spot Performance with (20% HS traffic)



(a) Pattern 1

(b) Pattern 2

(c) Pattern 3

(d) Pattern 4

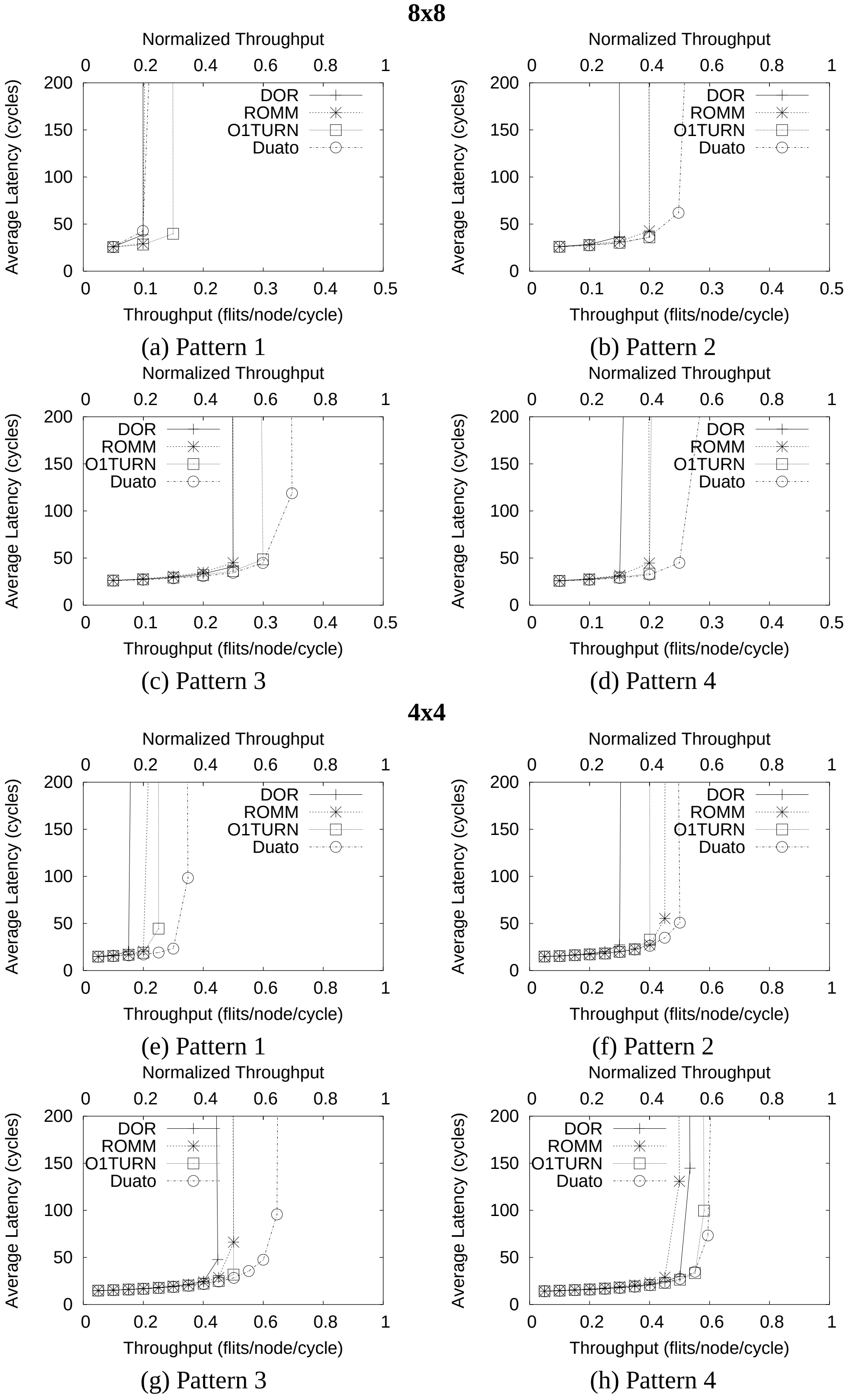
(e) Pattern 1

(f) Pattern 2

(g) Pattern 3

(h) Pattern 4

Figure 15: Hot Spot Performance with (30% HS traffic)



8x8

(a) Pattern 1

(b) Pattern 2

(c) Pattern 3

(d) Pattern 4

4x4

(e) Pattern 1

(f) Pattern 2

(g) Pattern 3

(h) Pattern 4

Figure 16: Hot Spot Performance with (40% HS traffic)