A New Worst-Case Throughput Bound for Oblivious Routing in Odd Radix Mesh Network

Guang Sun*, Chia-Wei Chang† and Bill Lin†
*Tsinghua University †University of California, San Diego

Abstract—$1/2$ network capacity is often believed to be the limit of worst-case throughput for mesh networks. However, this letter provides a new worst-case throughput bound, which is higher than $1/2$ network capacity, for odd radix two-dimensional mesh networks. In addition, we propose a routing algorithm called U2TURN that can achieve this worst-case throughput bound. U2TURN considers all routing paths with at most 2 turns and distributes the traffic loads uniformly in both X and Y dimensions. Theoretical analysis and simulation results show that U2TURN outperforms existing routing algorithms in worst-case throughput. Moreover, U2TURN achieves good average-throughput at the expense of approximately 1.5x minimal average hop count.

Index Terms—Networks-on-Chip (NoC), routing, worst-case throughput, average-case throughput, mesh.

1 Introduction

Networks-on-chip (NoC) has been proposed as a new intra-chip communication infrastructure. Regular topologies, especially the mesh topology, have been widely deployed [1][2]. The choice of oblivious routing algorithm, which is oblivious of the network state when determining a route, has a significant impact on the network performance, such as the throughput and the latency. Average-case throughput provides an average performance metric, whereas worst-case throughput provides a performance guarantee.

It is well-known and has been shown that the optimal worst-case throughput for oblivious routing in two-dimensional mesh networks is $1/2$ of the network capacity when the radix is *even*. For example, such an analysis for the even radix case is provided in [1].

For the *odd* radix case, it is also well-known that a worst-case throughput of $1/2$ network capacity can be achieved. This $1/2$ of network capacity is often believed to be the limit of the worst-case throughput. For example, in the near optimality analysis of the O1TURN routing algorithm [4], the near optimality of O1TURN in the odd radix case is relative to a presumed $1/2$ network capacity limit.

In this letter, we provide a new worst-case throughput bound of $(k+1)/(2k+1)$ of network capacity for the odd radix two-dimensional mesh with radix k, which is higher than $1/2$ network capacity. We propose a non-minimal oblivious routing algorithm called U2TURN (uniform 2-turn) that can achieve this worst-case throughput bound. Improved analytical results for oblivious routing are important because they often form the theoretical underpinnings for a variety of network analysis problems.

The main contributions of this letter are as follows:

(a) We provide a new worst-case throughput bound, which is higher than $1/2$ network capacity, for the odd radix two-dimensional mesh.

(b) U2TURN achieves this worst-case throughput. It outperforms VAL[10], DOR [5], ROMM [3] and O1TURN [4] in worst-case throughput for all odd radix mesh networks.

(c) For average-case throughput, U2TURN also outperforms VAL, DOR, O1TURN by 25.1%, 43.7% and 20.8%, respectively, on average for different network sizes evaluated.

The rest of this letter is organized as follows. Section 2 provides a brief background on routing algorithms and performance metrics. Section 3 describes our U2TURN routing algorithm and provides theoretical performance analysis. Section 4 provides simulation results. Finally, Section 5 draws conclusions.

2 Background

2.1 Routing Algorithms

Dimension-order routing (DOR) [5] is simple and achieves minimal-length routing. However, it suffers from poor worst-case and average-case throughput due to the lack of routing diversity. On the other hand, Valiant proposed a routing algorithm called VAL [10] that offers excellent worst-case throughput (half network capacity), but it suffers from poor average-case throughput and long routing path. ROMM [3] overcomes some of the drawbacks of DOR and VAL. It achieves minimal-length routing and good average-case throughput. However, ROMM has even poorer worst-case throughput than DOR [8]. Thus, these existing routing algorithms suffer from either poor worst-case throughput (e.g. DOR, ROMM) or poor latency (e.g. VAL).

O1TURN [4] achieves both minimal-length routing and good throughput. For 2D odd radix mesh networks, O1TURN has worst-case throughput within a factor of $1/k^2$ of half of network capacity. However, instead of using at most one turn in O1TURN routing, our U2TURN routing considers all routing paths with at most 2 turns and distributes the traffic uniformly in both X and Y dimensions, which results in higher worst-case throughput than half of network capacity for all odd radix mesh networks. Simulations show that U2TURN outperforms the above routing algorithms in both worst-case and average-case throughput.

Although the routing algorithms described in [9] and [7] also use routes with at most 2 turns, they are developed for torus networks rather than meshes. Besides, 2TURN

• *Manuscript submitted: 07-Dec-2011. Manuscript accepted: 25-Mar-2012. Final manuscript received: 28-Mar-2012.*

[9] does not have a closed-form description. It requires solving a linear program for each network radix. These linear programs grow rapidly, making them difficult to scale to large networks. IVAL [9] has only been shown to achieve the same worst-case throughput as VAL for torus networks, whereas U2TURN achieves a higher worst-case throughput than VAL in the case of odd radix mesh networks. Like I2TURN and W2TURN described in [7] for torus networks, U2TURN also provides closed-form analytical expressions for evaluating the average path length. These closed-form analytical expressions helped us to derive a routing algorithm that achieves a worst-case throughput better than the $1/2$ network capacity previously believed to be the limit of worst-case throughput for mesh networks.

2.2 Performance Metrics

In this section, we give the performance metrics, such as worst-case and average-case throughput, for oblivious routing algorithms. In order to simplify the discussion on these performance metrics, like [4] [6], we ignore flow control issues, and assume single flit packets that route from node to node in one cycle. In particular, we elaborate on the concept of network capacity, and provide a brief overview of analytical methods to evaluate worst-case and average-case throughput for oblivious routing algorithms.

Network Capacity: Network capacity is defined by the maximal sustainable channel load Υ^* when a network is loaded with uniformly distributed traffic [4]. For any k-ary n-mesh [1][6], independent of n,

$$\Upsilon^* = \begin{cases} \frac{k}{4}, & k \text{ is even} \\ \frac{k^2-1}{4k}, & k \text{ is odd} \end{cases} \tag{1}$$

where Υ^* is the inverse of the network capacity.

Worst-Case Throughput: For a given routing algorithm R and traffic matrix Λ, the maximal channel load $\Upsilon(R, \Lambda)$ is the expected traffic loads crossing the heaviest loaded channel under R and Λ [6]. For a given routing algorithm R, the worst-case channel load $\Upsilon_{wc}(R)$ is the heaviest channel load that can be caused by any admissible traffic [6]. A double sub-stochastic traffic matrix Λ is admissible when it has row and column sums of at most 1. For a given routing algorithm, [8] provided a method to find worst-case traffic that can cause worst-case channel load and worst-case throughput. Note that the worst-case channel load is the inverse of worst-case throughput. Like [6][4], we use the normalized worst-case throughput, which is normalized to the network capacity, as worst-case performance metric:

$$\Theta_{wc}(R) = \left(\frac{\Upsilon_{wc}(R)}{\Upsilon^*}\right)^{-1} \tag{2}$$

Average-Case Throughput: As for average performance metric, we use normalized average-case throughput. Using the same method in [6][4][8], the average-case throughput for a given routing algorithm R is found by averaging the throughput over a large set of random traffic matrix T (e.g. $|T| = 1000000$):

$$\Theta_{avg}(R) = \frac{1}{|T|} \sum_{\Lambda \in T} \left(\frac{\Upsilon(R, \Lambda)}{\Upsilon^*}\right)^{-1} \tag{3}$$

3 U2TURN Routing

3.1 Routing Description

As the name implies, U2TURN considers any routing path between the source and destination with *at most* 2 turns, where each turn changes dimension from X to Y or Y to X. Note that "U-turns", turns that reverse direction along the same dimension, are not allowed in U2TURN.

In comparison to O1TURN routing, U2TURN is more flexible in that O1TURN routes are a subset of U2TURN routes. O1TURN only uses routes with at most one turn. However, our U2TURN routing considers all routing paths with at most 2 turns and balances the traffic in both X and Y dimensions. The greater flexibility and more uniform traffic distribution enable U2TURN to achieve better throughput performance.

For implementation, U2TURN routing consists of two parts: XYX-Routing with 0.5 probability and YXY-Routing with also 0.5 probability.

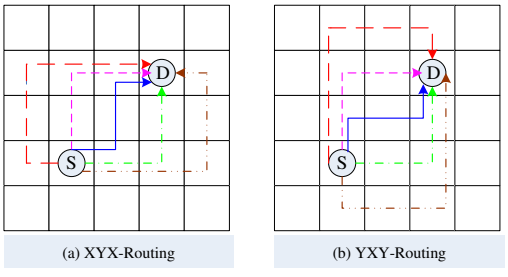
First, we elaborate on XYX-Routing. Suppose (x_1, y_1) is the source node and (x_2, y_2) is the destination node. XYX-Routing is a three-segment-routing generated as follows:

(a) X-segment: choose at uniform random an X position $x^* \in [0, k-1]$, and route in X-dimension from source node (x_1, y_1) to (x^*, y_1).

(b) Y-segment: route from (x^*, y_1) to (x^*, y_2) in Y-dimension.

(c) X-segment: route from (x^*, y_2) to destination node (x_2, y_2) in X-dimension.

Each of these three segments uses minimal-length routing, although the combination of segments can result in non-minimal routing along the entire path. There is a degenerate case. When $y_1 = y_2$, it just needs one-segment routing: route directly from (x_1, y_1) to (x_2, y_2) using minimal-length routing in the X-dimension. Fig. 1(a) depicts five possible XYX-Routing paths from source node to destination node for 5×5 mesh, where each path has a probability of $1/5$. YXY-Routing is similar to XYX-Routing for symmetry, which is shown in Fig. 1(b).



(a) XYX-Routing (b) YXY-Routing

Fig. 1. Routing paths for XYX-Routing and YXY-Routing in 5×5 mesh.

From the aforementioned routing description, we know that XYX-Routing and YXY-Routing cover all routing paths between the source and destination with at most 2 turns. In addition, XYX-Routing distributes the traffic uniformly along the X-Dimension, and YXY-Routing distributes the traffic uniformly along the Y-dimension. U2TURN uses XYX-Routing and YXY-Routing with equal probability, so that it balances traffic loads in both X and Y dimensions, which results in a high throughput.

As for the deadlock problem, two virtual channels per physical channel is sufficient to achieve deadlock-free routing by incrementing a packet's virtual channel set after each turn from the Y to the X dimension.

3.2 Performance Analysis

3.2.1 Throughput Analysis

In this section, we prove that our U2TURN achieves a new worst-case throughput bound of $(k+1)/(2k+1)$ of network capacity for an odd radix two-dimensional mesh with radix k, which is higher than $1/2$ of network capacity. We prove this by three claims as follows.

Claim 1. *For the odd radix one-dimensional mesh, the worst-case channel load of minimal-length routing is* $\frac{k-1}{2}$.

Proof: [8] proposed a method, which can find the worst-case traffic for any given oblivious routing algorithm in a given topology. So, we use this method, and find that the worst-case traffic for minimal-length routing in one dimension mesh is Tornado: from x to $(x + \frac{k-1}{2}) \bmod k$, where k is the radix. The heaviest channel load of the minimal-length routing under Tornado traffic matrix is $\frac{k-1}{2}$. Therefore, this is the worst-case channel load for minimal-length routing in odd radix one-dimensional mesh. As the worst-case throughput is the inverse of worst-case channel load, the worst-case throughput of minimal-length routing is $\frac{2}{k-1}$, which is higher than half of network capacity. □

Claim 2. *For the odd radix two-dimensional mesh, the worst-case channel load of XYX-Routing is:* $\frac{k-1}{2}$ *in Y-dimension and* $\frac{k^2-1}{2k}$ *in X-dimension, respectively.*

Proof: As discussed in Section 2.2, the maximal channel load under uniformly distributed traffic is $\frac{k^2-1}{4k}$ when k is odd. XYX-Routing consists of three segments routing, and the two X-segment routings get twice uniform. The maximal channel load under twice uniform traffic matrix is simply $\frac{2(k^2-1)}{4k} = \frac{k^2-1}{2k}$. Thus, for the odd radix two-dimensional mesh, the worst-case channel load of XYX-Routing in X-dimension is $\frac{k^2-1}{2k}$.

As for the Y-dimension case in XYX-Routing, it uses minimal-length routing from (x^*, y_1) to (x^*, y_2) . Since traffic is uniformly distributed along the X-dimension, the Y-dimension channels with different X positions but the same Y position have the same traffic loads. So the traffic of Y-dimension of XYX-Routing is degenerated into the case in Claim 1, namely, minimal-length routing in one-dimension mesh. Thus, we get that the maximal channel load in Y dimension for XYX-Routing is $\frac{k-1}{2} < \frac{k^2-1}{2k}$.

Therefore, the worst-case channel load for XYX-Routing is $\frac{k^2-1}{2k}$. And the worst-case throughput for XYX-Routing is $\frac{2k}{k^2-1}$. As shown in Section 2.2, the network capacity for the odd radix mesh is the inverse of Υ^*, namely, $\frac{4k}{k^2-1}$. Thus, the worst-case throughput for XYX-Routing equals to half of network capacity. □

Claim 3. *For odd radix two-dimensional mesh, U2TURN achieves a worst-case throughput bound of* $(k+1)/(2k+1)$ *of network capacity, which is higher than half of network capacity.*

Proof: The worst-case channel load analysis for YXY-Routing is the same with XYX-Routing because of symmetry. Similarly, it follows that the worst-case channel load of YXY-Routing in odd radix two-dimensional mesh is $\frac{k-1}{2}$ in X-dimension and $\frac{k^2-1}{2k}$ in the Y-dimension, respectively.

U2TURN consists of two parts: XYX-Routing with 0.5 probability and YXY-Routing with also 0.5 probability. Thus, the worst-case channel load of U2TURN in X-dimension is $0.5(\frac{k-1}{2}) + 0.5(\frac{k^2-1}{2k}) = \frac{2k^2-k-1}{4k}$. Similarly, the worst-case channel load of U2TURN in Y-dimension is also $\frac{2k^2-k-1}{4k}$. So, the the worst-case channel load of U2TURN is $\frac{2k^2-k-1}{4k}$ and the worst-case throughput is $\frac{4k}{2k^2-k-1}$.

As shown in Section 2.2, the network capacity for odd radix mesh is $\frac{4k}{k^2-1}$. Therefore, for an odd radix two-dimensional mesh, U2TURN achieves a worst-case throughput bound of $\left(\frac{4k}{2k^2-k-1}\right) / \left(\frac{4k}{k^2-1}\right) = (k+1)/(2k+1)$ of network capacity, which is higher than half of network capacity for all radix k. □

Fig. 2 shows that U2TURN outperforms VAL, DOR, ROMM and O1TURN in worst-case throughput for different network sizes.

Different with U2TURN, the optimal routing (including 2TURN) proposed in [9] does not have a closed-form description, which requires solving a separate linear program for each network radix. These linear programs grow so fast that it is difficult to scale to large networks [7].

Note that U2TURN achieves the same optimal worst-case throughput as the optimal routing that can be obtained with the method proposed in [9].

The worst-case throughput of U2TURN is also optimal for even radix k, which is $1/2$ network capacity, as with VAL and O1TURN. The proof for even case is a direct extension of the proof of odd case and is not presented here due to a page limit.

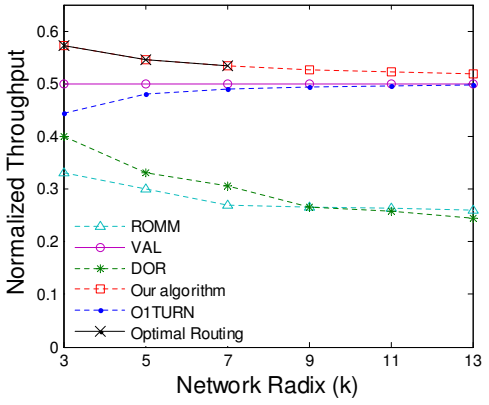


Fig. 2. Worst-case throughput for odd radix mesh.

3.2.2 Latency Analysis

For latency analysis, we use the average hop count $H_{ave}(R)$ for a given routing algorithm R to measure the average latency. We assume that the traffic between all source-destination pairs are all the same when computing $H_{ave}(R)$ [6]. The average hop count for DOR in k-ary 2-mesh network is $2\left(\frac{k^2-1}{3k}\right)$, where each $\frac{k^2-1}{3k}$ component represents the average hop count in each dimension using minimal-length routing [1][6].

In U2TURN, minimal-length routing is used in three segments. However, as mentioned in Section 3.1, there is

a degenerated case when $y_1 = y_2$, which has a probability of $1/k$. In this degenerated case, there is only one minimal-length routing segment. Therefore, the average hop count of U2TURN is given as follows:

$$H_{ave}(U2TURN) = \frac{k-1}{k} \times 3(\frac{k^2-1}{3k}) + \frac{1}{k} \times \frac{k^2-1}{3k} = \frac{3k-2}{k}(\frac{k^2-1}{3k}) \quad (4)$$

In order to quantify the penalty factor in average latency for given routing algorithm R over DOR, we normalize the average hop count of R to DOR. So the penalty factor for U2TURN is $\frac{3k-2}{2k}$, which is less than 1.5 for all radix k.

4 Evaluation

In this section, we compare the throughput of U2TURN with VAL, DOR and O1TURN with respect to each traffic matrix shown in Table 1. Table 2 provides the evaluation results for different network sizes. As shown in Table 2, U2TURN outperforms all other routing algorithms in worst-case throughput and achieves a new optimal worst-case throughput bound of $(k+1)/(2k+1)$, which is higher than $1/2$ of network capacity. This also validates the throughput analysis of U2TURN in Section 3.2.1. For average-case throughput, U2TURN also outperforms VAL, DOR, and O1TURN, on average over the three evaluated network sizes, by 25.1%, 43.7% and 20.8%, respectively. U2TURN also performs very well in adversarial traffic matrix, namely, Transpose and DOR-WC. For Complement and Nearest-Neighbor traffic, U2TURN outperforms VAL, but does worse than DOR and O1TURN. For ideal uniformly Random traffic, although DOR achieves better throughput than U2TURN, DOR is significantly worse in the average and worst case. In summary, our evaluation shows that U2TURN performs very well in different network sizes and traffic matrices.

TABLE 1
Traffic Matrices Evaluated.

Worst-Case	Worst-case traffic matrix that results in lowest throughput
Average-Case	Average throughput over a million random traffic matrix
Transpose	Packets route from (x,y) to (y,x)
Random	Destination node chosen at uniform random
DOR-WC	Worst-case traffic for DOR. Packets route from (x,y) to (k-y-1,k-x-1)
Complement	Packet at (x,y) sent to (k-x-1,k-y-1).
Nearest-Neighbor	Packets sent to nodes one hop away with equal probability

5 Conclusion

U2TURN is a closed-form non-minimal oblivious routing algorithm for odd radix two-dimensional mesh network. We prove it achieves a new worst-case throughput bound of $(k+1)/(2k+1)$ of network capacity with radix k, which is higher than $1/2$ network capacity. Thus, it outperforms existing algorithms such as VAL, DOR, ROMM and O1TURN in worst-case throughput. In addition, when compared with the optimal routings that can be obtained with the method proposed in [9], U2TURN can achieve the same optimal worst-case throughput. For average-case throughput, U2TURN also outperforms VAL, DOR and O1TURN on average of three simulation cases by 25.1%, 43.7% and 20.8%, respectively. Moreover, we provide a closed-form expression for the average hop count of U2TURN in latency analysis. In comparison with DOR and O1TURN, U2TURN achieves a higher throughput at the expense of an average hop count that is approximately 1.5x minimal.

TABLE 2
Normalized Worst-Case & Average-case Throughput

	3x3 mesh			
	VAL	DOR	O1TURN	U2TURN
Worst-case	0.5	0.33	0.44	0.57
Average-case	0.5	0.405	0.477	0.604
Transpose	0.5	0.33	0.67	0.80
Random	0.5	1	1	0.72
DOR-WC	0.5	0.33	0.67	0.80
Complement	0.5	0.67	0.67	0.57
Nearest-Neighbor	0.5	1.33	1.33	0.75
	5x5 mesh			
	VAL	DOR	O1TURN	U2TURN
Worst-case	0.5	0.3	0.48	0.55
Average-case	0.5	0.441	0.529	0.632
Transpose	0.5	0.3	0.6	0.75
Random	0.5	1	1	0.685
DOR-WC	0.5	0.3	0.6	0.75
Complement	0.5	0.6	0.6	0.55
Nearest-Neighbor	0.5	2.4	2.4	1.17
	7x7 mesh			
	VAL	DOR	O1TURN	U2TURN
Worst-case	0.5	0.286	0.49	0.53
Average-case	0.5	0.461	0.550	0.640
Transpose	0.5	0.286	0.57	0.73
Random	0.5	1	1	0.686
DOR-WC	0.5	0.286	0.57	0.73
Complement	0.5	0.57	0.57	0.533
Nearest-Neighbor	0.5	3.4	3.4	1.32

References

[1] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[2] S. Guang, L. Shijun, J. Depeng, L. Yong, S. Li, Y. Zhang, and Z. Lieguang. Performance-aware hybrid algorithm for mapping ips onto mesh-based network on chip. *IEICE TRANSACTIONS on Information and Systems*, 94(5):1000–1007, 2011.

[3] T. Nesson and S. 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, 1995.

[4] D. Seo, A. Ali, W. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 432–443. IEEE Computer Society, 2005.

[5] H. Sullivan and T. Bashkow. A large scale, homogeneous, fully distributed parallel machine, i. In *ACM SIGARCH Computer Architecture News*, volume 5, pages 105–117. ACM, 1977.

[6] R. Sunkam Ramanujam and B. Lin. Randomized partially-minimal routing on three-dimensional mesh networks. *Computer Architecture Letters*, 7(2):37–40, 2008.

[7] R. Sunkam Ramanujam and B. Lin. Weighted random oblivious routing on torus networks. In *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 104–112. ACM, 2009.

[8] B. Towles and W. Dally. Worst-case traffic for oblivious routing functions. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–8. ACM, 2002.

[9] B. Towles, W. 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, 2003.

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