A Scalable Routing Methodology for Low-Latency Interconnection Networks

Ryuta Kawano

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

School of Science for Open and Environmental Systems
Graduate School of Science and Technology
Keio University

December 2017

Preface

End-to-end network latency has become an important issue for parallel application on large-scale High Performance Computing (HPC) systems. For large parallel applications executed on the next generation of HPC systems, MPI communication latency should be lower than one microsecond. Switch delays (e.g., about 100 nanoseconds in InfiniBand QDR) are typically larger than the wire and flit injection delays, even when including serial and parallel converters. This is why inter-switch topologies should have a low diameter and low average shortest path length, both of which can be measured in terms of a number of switch hops.

In order to cope with this requirement, recently researchers have developed random topologies applicable for inter-switch networks. Compared with the conventional Torus or Fat-tree networks, these topologies can drastically reduce the number of hops and be efficiently applied to HPC systems, data centers, and many-core systems.

These irregular topologies often suffer from their lack of scalability for feasible interconnection networks. Firstly, they tend to increase the total amount of cable length between cabinets and thus to increase the cost. Secondly, the number of table entries consumed on each switch is increased. Thirdly, they increase the time complexity to compute assignment of Virtual Channels (VCs) to the routing paths for deadlock-freedom.

To improve their scalability, this thesis focuses attention on layout-conscious random topologies that contain randomly connected links with length limitation. These networks achieve drastic reduction of the total cable length with a minimal increase in the number of hops compared with completely random networks.

This thesis aims to explore a scalable routing methodology that can be applied to the layout-conscious random topologies to achieve feasible low-latency interconnection networks. Firstly, a scalable routing method for the layout-conscious random topologies, LOREN (Layout-Oriented Routing with Entries for Neighbors), is proposed. This method exploits irregularity and locality in these topologies to achieve both the small number of hops between nodes and small routing table sizes required. Secondly, an advanced method for the VC assignment, ACRO (Assignment of Channels in Reverse Order), is proposed. This approach has a small time complexity, yet with the same number of VCs compared with conventional VC assignment methods.

The proposed routing methodology provides the better trade-offs between an achieved network latency and the number of required table entries. It is evaluated by using a cycle-accurate network simulator. In the evaluations, the proposed routing method LOREN is enhanced by the proposed VC assignment method ACRO to guarantee the deadlock-freedom. The results show that the proposed methodology can improve the network throughput by up to 67.9 % compared to a conventional compact routing method. Moreover, the number of required routing table entries is reduced by up to 91 %, which improves scalability and flexibility for implementation.

Acknowledgments

There are a number of people without whom my doctoral study might not have been accomplished, and to whom I am greatly indebted.

Most of all, I would like to express my deepest gratitude to my supervisor, Professor Hideharu Amano. His constant, patient and generous guidance, stimulating suggestions and hopeful encouragements have been my endless source to pursue my work for the six years spending in his laboratory. He has always given me depthful words whenever I had a difficulty in my research life, and reviewed every paper that I have written with constructive comments.

This thesis has not been possible without elaborate, extensive and integral comments by my doctoral committee members, Professor Hiroaki Yamanaka, Associate Professor Hiroki Matsutani, and Associate Professor Takahiro Yakoh. They spared their valuable year-end and new-year holidays to review my thesis, and pointed out the shortcomings of my work with mighty hearts.

I am greatly indebted to Associate Professor Michihiro Koibuchi at National Institute of Informatics, who has been my primary collaborator, and at the same time, a symbol of my admiration as a researcher.

I would like to thank all my BlackBus group members. I thank all h_superusers and all the “hlab” members for offering daily inspirations and so much fun for me. Last but not least, I am obliged to our secretary Kazuko Fujita for her careful and firm stewardship.

Ryuta Kawano
Yokohama, Japan
November 2017

謝辞

博士論文完成までの長い大学院生活の間、精神的に不安定な時期も含め、献身的にサポートしてくれた家族に深く感謝致します。

神奈川の大学に通う私を遠いところから気にかけてくれた、父方・母方それぞれの祖父母に深く感謝致します。
また、私のことを厳しくも優しく見守り、大学院生活の間は同じ家に住み、私の博士論文完成を最後まで見届けてくれた父・一隆に深く感謝致します。

そして、私を生み、育ててくれた母・珠美への感謝は、言葉で言い尽くすことが出来ません。博士課程在籍中に研究がうまくいかず荒んでいた時もずっと心配してくれ、立ち直るきっかけを与えてくれました。今年10月に博士論文完成までの目途をつけたことを報告した際は、とても喜んでくれていたのを覚えています。

2017年11月1日、母は広島の実家で倒れ、私の博士論文の完成を最後まで見届けることなく、53年の生涯を閉じました。私が生まれてからのおよそ28年間、母・珠美が生前に与えてくれた深い愛情に改めて最大限の感謝を述べるとともに、この博士論文を捧げます。

2017年11月13日
横浜にて
河野 隆太

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, the subject of this paper, the necessity of low-latency interconnection networks for HPC systems is introduced. Moreover, I overview the objectives and contribution of this paper. The organization of this paper is also shown.

1.1 Background

With improving parallelism and increasing scientific computing or big data analytics, low-latency interconnection networks are essential as well as processing performance of computational nodes for modern computing systems [2, 3]. It is generally agreed today that a high-performance and a large-scale computing system called exascale computing will be realized by the next-generation dedicated parallel computers. It is also considered to use a tremendous computing system called a warehouse-scale computer for big data processing. These parallel computers usually become large scales from about 50,000 to 100,000 hosts (computing nodes). In these large systems, the time consumed by the computation cannot efficiently hide the time consumed by exchanging data on the networks, which drastically degrades the system performance.

To reduce the network latency, network topologies with the low diameter and the low average path length can be efficiently applied to interconnection networks. The performance of off-chip interconnection networks is usually dominated with the delay in switching fabrics (e.g., about a hundred nanoseconds in Infiniband QDR) rather than in link and injection. Moreover, Networks-on-Chip (NoCs) that are used for many-core architectures increase the latency for transferring short-length packets even when using conventional wormhole-switching routers. Therefore, researchers have recently focused attention on low-latency networks with high-radix switches, which can be modeled as small-diameter topologies with large degrees [4–6].

Recently proposed irregular topologies adopted in inter-switch networks can significantly reduce the latencies between computational nodes while can achieve flexible network structures [7–9], while the traditional structured network topologies with either low-radix or high-radix switches cannot. Moreover, small-diameter topologies for arbitrary network sizes are known to usually have irregular structures [10]. These networks can contribute to reduce the latency by exploiting *small-world* phenomena [11], and thus to improve the performance of parallel computation not only for off-chip networks but for on-chip inter-core networks with low-radix routers [12, 13].

A lately proposed more scalable solution generates random shortcut links with their lengths limited [14, 15]. Unlike conventional non-random topologies, fully random topologies have a drawback of an increased link length. This causes both an increased aggregate cable length on a machine room floor for HPC systems and frequency reduction due to an increased critical path on NoCs (Networks-on-Chip) for many-core systems. It is reported that the layout-conscious topologies achieve a comparable average number of hops to completely irregular topologies yet reduce the total link length.

1.2 Objectives

There are several challenges to achieve practical irregular networks. The main objective of this thesis is to provide a feasible solution to introduce the randomized topologies for practical HPC systems and many-core systems.

Firstly, conventional routing algorithms for non-random networks cannot be utilized in random networks. It is thus necessary for them to use topology-agnostic routing algorithms [16]. In these algorithms, each switch must have routing table entries for all destination switches. These table entries enable the optimal routing with the memory space of $O(|N| \cdot \log |N|)$ for each switch. Due to this large memory space in routing tables, the conventional algorithms for irregular networks degrade scalability for larger system sizes.

Secondly, routing methods for irregular networks do not naively guarantee deadlock-freedom that has to be supported both on practical HPC networks and NoCs. Therefore, endorsing a given livelock-free routing with deadlock-freedom is needed.

1.3 Contribution

This thesis explores a scalable routing methodology that is suitable for the layout-conscious random topologies to achieve feasible low-latency interconnection networks for HPC systems, data centers, and many-core architectures.

Firstly a routing method for layout-conscious random topologies LOREN (Layout-Oriented Routing with Entries for Neighbors) is introduced, which exploits irregularity and locality in these topologies to achieve both the small number of hops between nodes and small routing table sizes required.

Moreover, the proposed method called ACRO (Assignment of Channels in Reverse Order) exploits multiple Virtual Channels (VCs) [17, 18] for each physical channel in order to ensure LOREN deadlock-freedom. This approach has a small time complexity, yet with the same number of VCs compared to conventional VC assignment methods. Note that this method is also applicable for arbitrary routing methods. The proposed methodology considers a given topology and the routing table, obtained from a livelock-free routing algorithm, to generate the VC assignment to paths.

1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 surveys current interconnection network technologies including randomized topologies. Chapter 3 surveys a scalable routing methodology called *compact routing*, and topology-agnostic deadlock-free routing techniques. Chapter 4 proposes LOREN, a scalable and low-latency routing method for the layout-conscious random topologies. Chapter 5 proposes ACRO, an efficient VC assignment method with small time complexity. Chapter 6 studies network performance of LOREN with ACRO. This study models end-to-end latency and throughput of layout-conscious random topologies. Chapter 7 summarizes and discusses the above studies, and then concludes this thesis.

Chapter 2

Low-latency Network Topologies

This chapter surveys current interconnection network technologies including randomized topologies.

2.1 Conventional Regular and Irregular Topologies

State-of-the-art high-performance computing systems utilize well-structured and thus scalable network topologies. Table 2.1 shows the latest ranking of supercomputers and their network topologies [1]. As shown in this table, fat-tree, torus (*k*-ary *n*-cube), and dragonfly topologies are widely used in the recent systems.

Fat-tree and torus (*k*-ary *n*-cube) topologies are traditionally used for interconnection networks because the degree (the number of ports) is relatively small and thus low-radix switches can be utilized. Minimal routing can be achieved on these topologies with a few number of virtual channels for deadlock avoidance and the performance.

The recently proposed dragonfly topology [19] achieves the lower latency by exploiting high-radix switches that were realized thanks to the recent exponential increase in the pin bandwidth. This topology separates the whole network topology into inter-cabinet and intra-cabinet connections. Figure 2.3 shows a possible implementation that includes four cabinets, denoted as dotted squares, containing four hosts denoted as solid squares. In this example, the nodes within each cabinet are fully connected to improve the performance of the local traffic. Moreover, inter-cabinet connections can be implemented with distant links to improve the end-to-end network latency.

To achieve networks with the lowest number of hops, researchers in the field of graph theory have had interests in the concept of Moore graphs [20]. They achieve the optimum number of nodes in a graph with fixed degree and diameter. Hoffman-Singleton graph [21], which is one of the well-known Moore graphs, is exploited in intra-server networks [5]. Moreover, McKay-Miller-Širáň (MMS) graph [22] whose number of nodes is close to those of Moore graphs is utilized in Slim Fly topologies [6].

The problem of finding the largest graphs for given degree and diameter constraints has been well studied for decades. Thanks to the effort to approach optimal bounds, several graphs are found [23].

Table 2.1: Top 10 supercomputers and their network topologies [1].

Rank	Name	Network Topology
1	Sunway TaihuLight	Fat-tree
2	Tianhe-2	Fat-tree
3	Piz Daint	Dragonfly
4	Titan	Torus
5	Sequoia	Torus
6	Cori	Dragonfly
7	Oakforest-PACS	Fat-tree
8	K computer	Torus
9	Mira	Torus
10	Trinity	Dragonfly

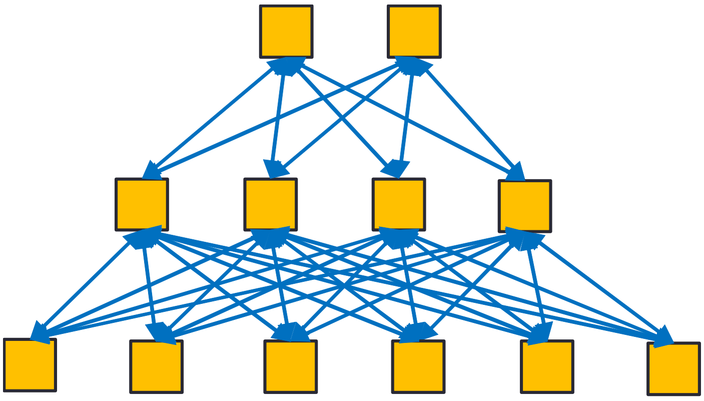


Figure 2.1: Fat-tree topology.

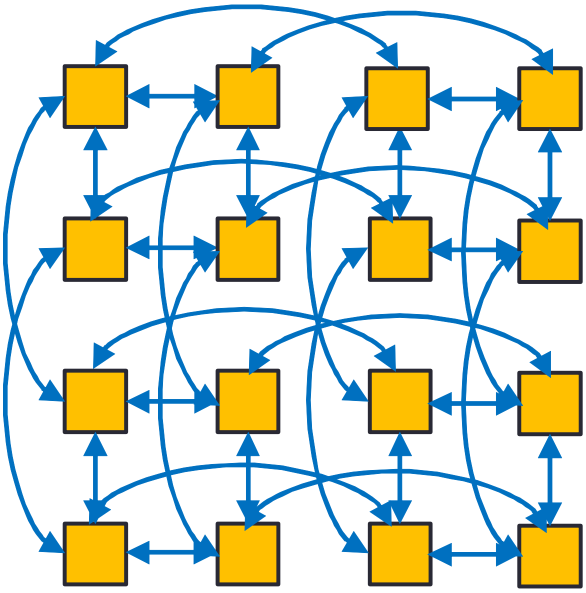


Figure 2.2: Torus (k-ary n-cube) topology.

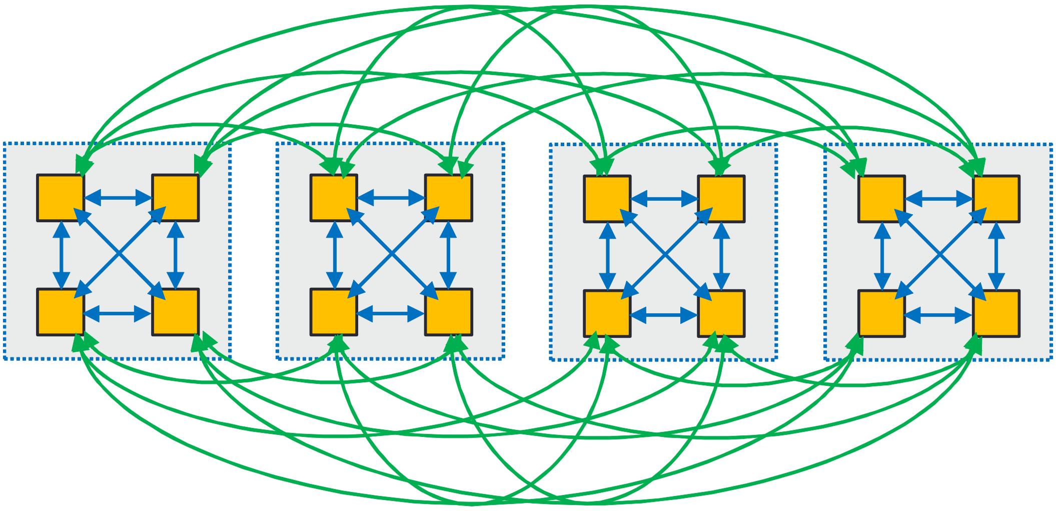


Figure 2.3: Dragonfly topology.

Although these graphs can drastically reduce the number of hops, they cannot be directly utilized with arbitrary numbers of nodes and degrees. This is because they are customized to specific numbers of nodes, whereas the numbers of nodes in practical HPC systems are decided depending on realistic plans, e.g., budget or application to be executed. Optimal topologies with given numbers of nodes and degrees have been explored, which reveals the difficulty in the optimization that is in most cases based on heuristic approach [10].

As a workaround plan against this difficulty, randomized topologies for HPC systems are proposed, which can achieve low latencies that are comparable with those in the optimized networks based on the concept of Moore graphs. Moreover, for arbitrary network sizes, randomized networks can be generated with the smaller computational costs than the optimized networks. These topologies can improve bandwidth, scalability, and fault-tolerance for both HPC networks [7–9] and on-chip networks [12].

2.2 Layout-conscious Random Topologies

Although random topologies achieve small numbers of hops between nodes, they need a larger amount of the total cable length, which degrades probability of implementation for practical inter-switch networks. This problem can be solved with networks which contain randomly connected links with length limitation [14,15]. These networks achieve drastic reduction of the total cable length with the minimal increase in the number of hops compared with completely random networks.

Limitation of the maximum link length also reduces the number of ports in each switch that achieves the optimum average number of hops. Let me assume a topology containing a set of nodes, N, developed on a 2-dimensional space. When the maximum link length L is not limited, the number of node pairs that can be connected with each other becomes $\frac{|N|}{2}$. On the other hand, the link length limitation reduces the number of connectable pairs approximately by $O(L^2)$. Since the order of the total link length can be calculated by the product of the number of links by the maximum link length, the length limitation significantly reduces the total link length. It is reported that the sufficient degree K (the number of ports) to achieve the optimum number of hops for the given maximum link length L can be calculated by the following formula [24].

$$K = \exp\left(\frac{L \cdot \log|N|}{\sqrt{|N|}}\right)$$

Figure 2.4 shows that the order of the sufficient degree is exponentially reduced as the given maximum link length decreases.

Figure 2.5 shows that the layout-conscious random topologies, indicated as LCR, achieves the low latency yet reduces the total cable length by random shortcut links with their length limited, compared to the conventional topologies described in Sec. 2.1. Traditional low-radix torus and fat-tree topologies reduce the total cable length yet achieve the worse latency. High-radix dragonfly and Slim Fly topologies achieve the low and optimal latency respectively while they consume a huge amount of the total cable length. A fully random topology denoted as Random can also achieve the quasi-optimal low latency with many distant shortcut links.

In this paper, all edges in the random topologies with the link length limited that are developed on a 2-dimensional surface are connecting between two nodes so that the Manhattan Distance between them, which is defined in Sec. 4.1.1, is equal or less than the limitation value r. Figure 2.6 shows an example of a layout-conscious topology that is placed on a 4×4 2-dimensional surface. The number of nodes $|N|$, the degree of each node d, and the maximum length of edges r are set to 16, 4, and 2, respectively. The label of a node, i, is defined with the equality $i = x_i + y_i \cdot n$, where (x_i, y_i) denotes a coordinate of the node i.

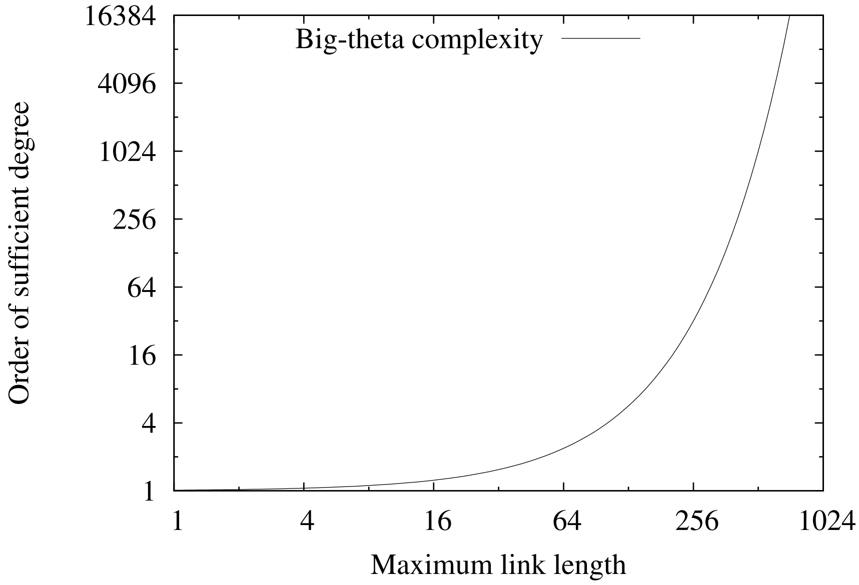


Figure 2.4: Order of sufficient degree for given maximum link length (1M nodes).

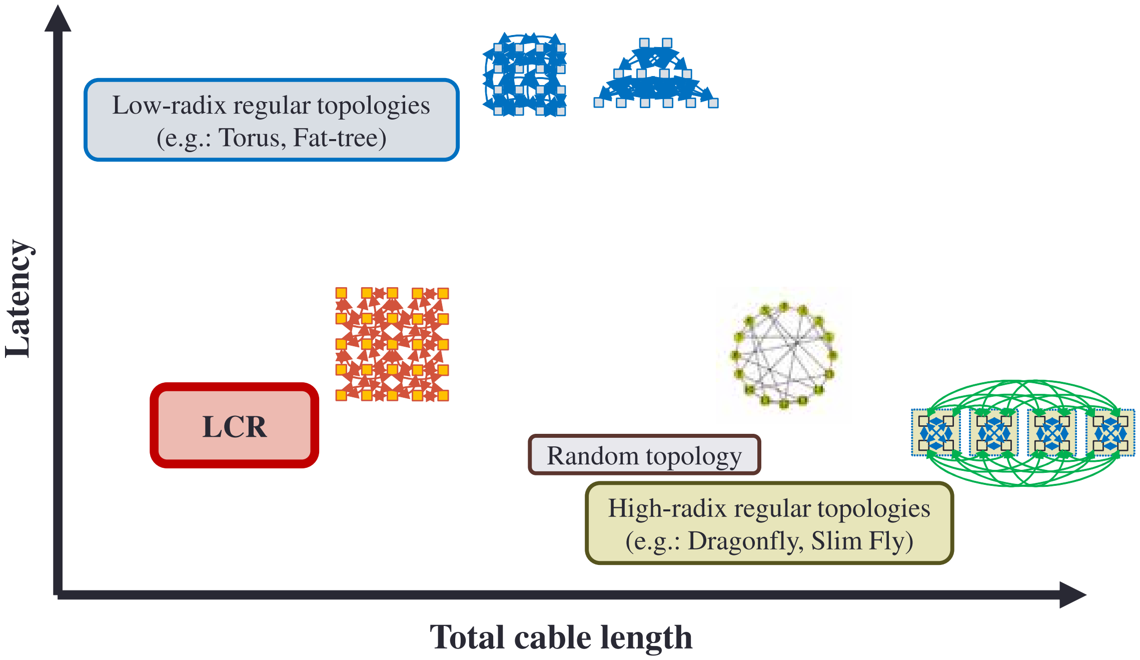

Low-radix regular topologies
(e.g.: Torus, Fat-tree)
Latency
LCR
Random topology
High-radix regular topologies
(e.g.: Dragonfly, Slim Fly)
Total cable length

Figure 2.5: Qualitative comparison of network topologies.

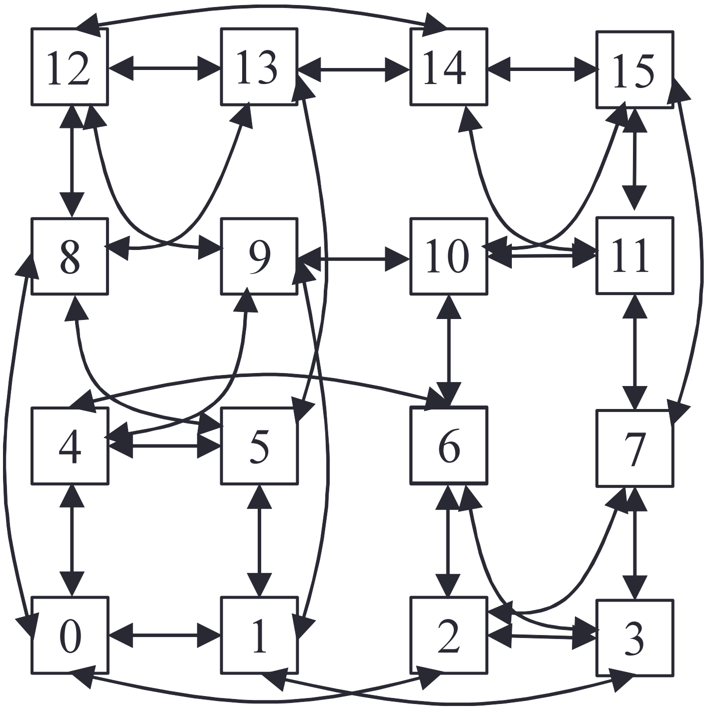


Figure 2.6: Example of layout-conscious random topology ($|N| = 16, d = 4, r = 2$).

Chapter 3

Routing Techniques

Traditional and recent regular topologies introduced in Sec. 2.1 can achieve minimal and deadlock-free routing without global information about their structures. Simple node labeling enables dimension order routing on torus topologies and up*/down* routing on fat-tree topologies. It also achieves minimal routing on dragonfly and Slim Fly topologies.

On the other hand, it is necessary for both fully and layout-conscious random topologies to use topology-agnostic routing algorithms [16]. We have to choose the appropriate algorithm for given topologies based on the two characteristics: a required routing table size (Sec. 3.1) and the difficulty in implementation for deadlock-freedom (Sec. 3.2).

3.1 Reduction of Routing Table Size

Both fully and layout-conscious random topologies require a routing table indicating the output link for all destination nodes in every router. If the number of nodes is more than thousands, the time to refer a certain size of memory for the table might bottleneck the operational frequency.

Moreover, the increased number of nodes degrades the possibility of the implementation with commodity network technologies for HPC systems. For example, on Infiniband networks, a packet only can include the information of a destination node as a destination LID (Local IDentifier) in the header [25]. Including all nodes in table entries will lack scalability because one sub-net can use approximately 48,000 LIDs at maximum. As shown in Fig. 3.1, the next generation HPC systems with randomized interconnection networks will obviously suffer from the increased number of computing nodes.

Researchers in the field of distributed computing have well discussed a methodology called compact routing, which is designed to reduce the size of required routing information up to a sublinear value at the cost of the minimum increase in numbers of hops. One of them [26] can be applied to power law graphs whose degree distribution follows a power law. This work assumes random regular graphs in which each node has the same degree. Thus, this method cannot be applied. Another method called Cowen's compact routing [27] uses topology-dependent names for nodes, and

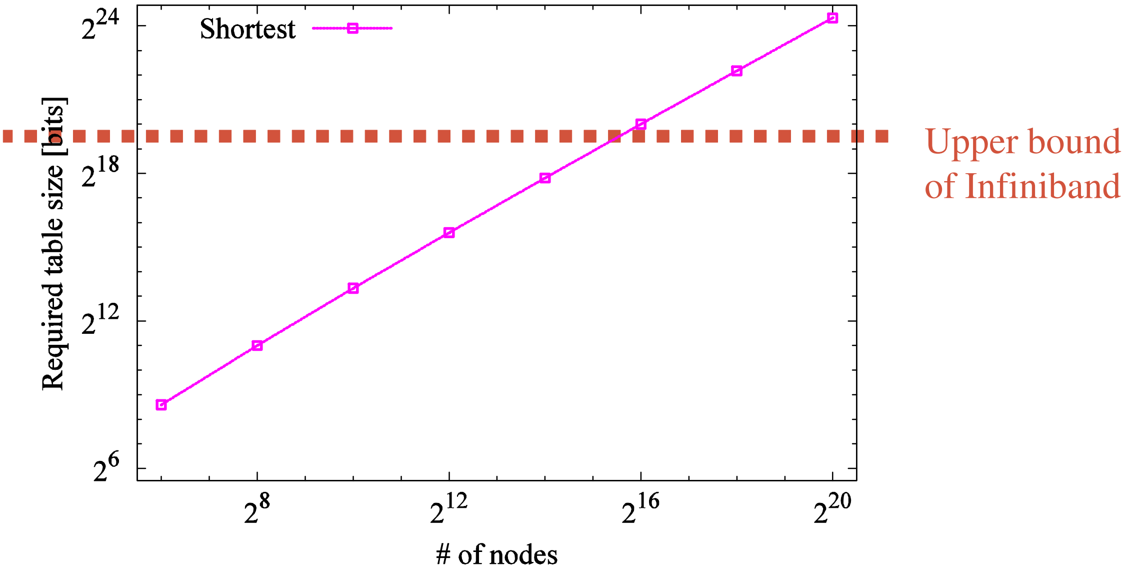
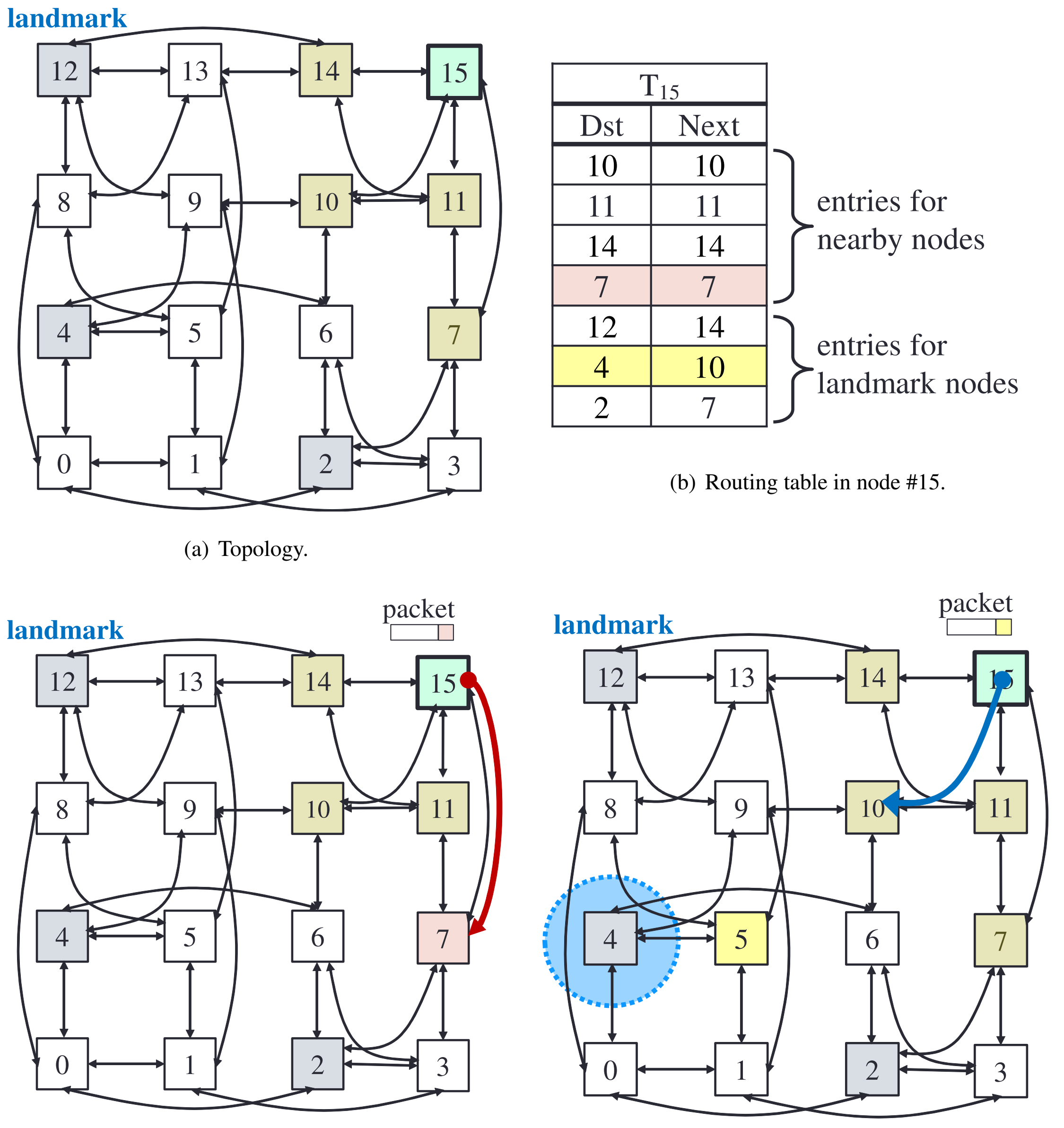


Figure 3.1: Routing table size required for shortest path routing in randomized networks and the upper bound for Infiniband.

achieves the maximum rate of increase in numbers of hops less than 3 with a local routing table of size $O(|N|^{2/3}\log^{4/3}|N|)$. The size of the table is reduced to $O(|N|^{1/2}\log^{3/2}|N|)$ by the improved implementation called Thorup's compact routing method. [28], which proves to be optimal.

With compact routing methods, some nodes are selected deterministically from a given network and are set as "landmarks" that cover all nodes. When a current node that a packet exists in is equal to a "landmark" node for the destination, it is forwarded to the next hop that is contained in a header of the packet and uses the minimal path towards the destination node. When the previous condition is not satisfied and there is an entry for a destination node of the packet in a routing table of the current node, the packet is forwarded based on the information. When these two conditions are not satisfied, which means the destination node is distant from the current node, firstly a landmark node for the destination node contained in a header of the packet is referred. Routing information for the landmark node in a routing table of the current node is then referred and the packet is forwarded based on the information.

An example of routing packets with compact routing methods is shown in Fig. 3.2. Figure 3.2(a) and 3.2(b) show a given topology and generated table entries in a node #15, respectively. In Fig. 3.2(b), each of the upper four entries contains the next hop for the shortest path towards the destination node. Furthermore, each of the lower three entries contains the next hop for the shortest path towards one of the landmark nodes. Let a packet be routed from the node #15 to #7. Since the entry for #7 is contained in the table, the packet is directly forwarded to #7, as shown in Fig. 3.2(c). On the other hand, when a packet whose destination node is #5 is routed on the node #15, the entry for the landmark node #4 that is nearest to #5 is referred. The packet is thus forwarded to a node #10, as shown in Fig. 3.2(d).



(a) Topology.

T15	
Dst	Next
10	10
11	11
14	14
7	7
12	14
4	10
2	7

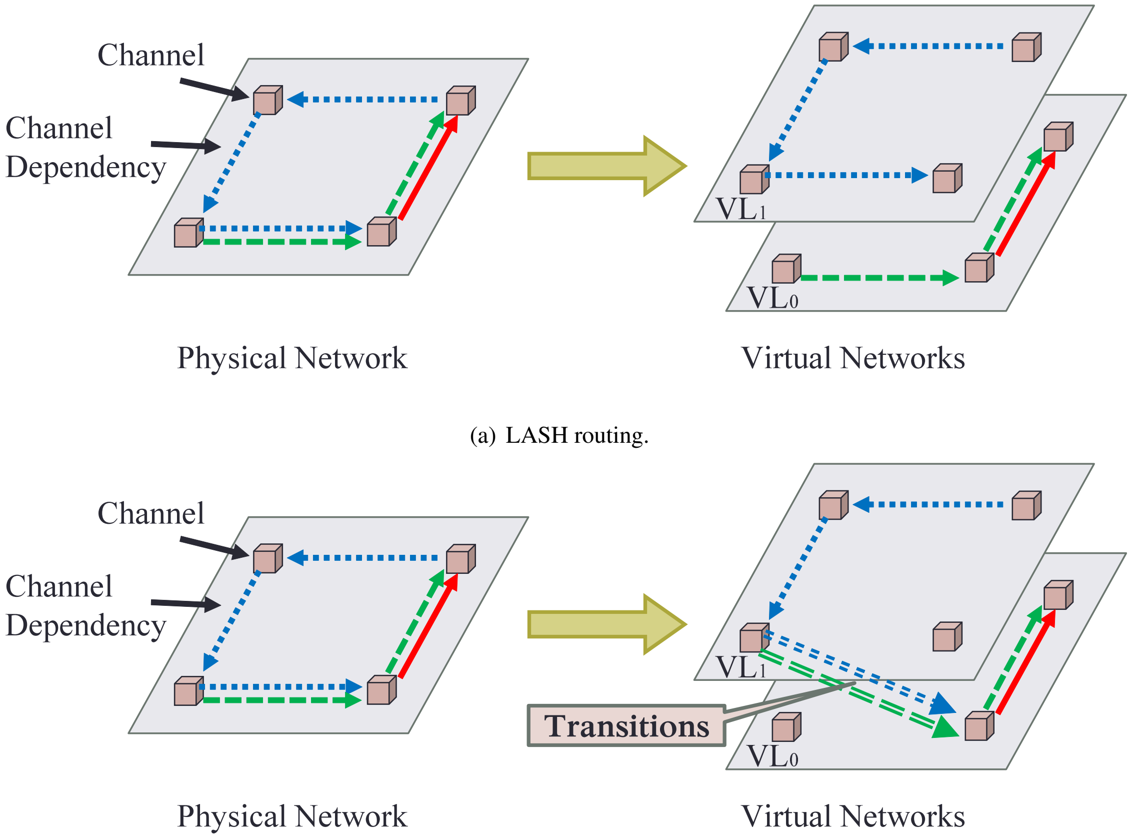
(b) Routing table in node #15.

(c) Routing towards nearby dst. node.

(d) Routing towards nearest landmark to distant dst. node.

Figure 3.2: Example of compact routing methods.

In summary, with compact routing methods, a minimal path is selected in the case of nearby destination nodes, otherwise, a path to one of the landmarks is selected to get close to the distant destination nodes. This routing method can reduce the amount of routing information with suppressing the increase of hops. However, the overall performance may be degraded because of the traffic concentration around landmarks.



(a) LASH routing.

(b) LASH-TOR routing.

Figure 3.3: Routings with virtual layers.

Table 3.1: Trade-offs on conventional algorithms.

Routing	# of required VLs	Time Complexity
LASH	9 for 256 SWs [18]	$O(\|N\|^2)$ [29]
LASH-TOR	4 for 256 SWs [18]	$O(\|N\|^3)$ [18]

3.2 Methodologies for Designing Deadlock-Free Routing

For irregular networks, the routing methods for regular topologies such as dimension order routing (DOR) for k-ary n-cubes are not applicable. It is thus necessary to use topology-agnostic routing methods for those networks [16]. There are some trade-offs on each routing method for arbitrary topologies. Up*/down* routing is commonly used and easily implementable for large irregular networks. Its ease of implementation comes from the required number of Virtual Channels (VCs). With the assumption of using one node as a root of the spanning tree, up*/down* routing supports deadlock-freedom with only one VC. The weakness of this routing is that it does not support minimal paths for most of traffics.

Conventional deadlock-free virtual channel assignment methods, LASH and LASH-TOR routing techniques, can assign each path between source and destination nodes to Virtual Layers (VLs), each

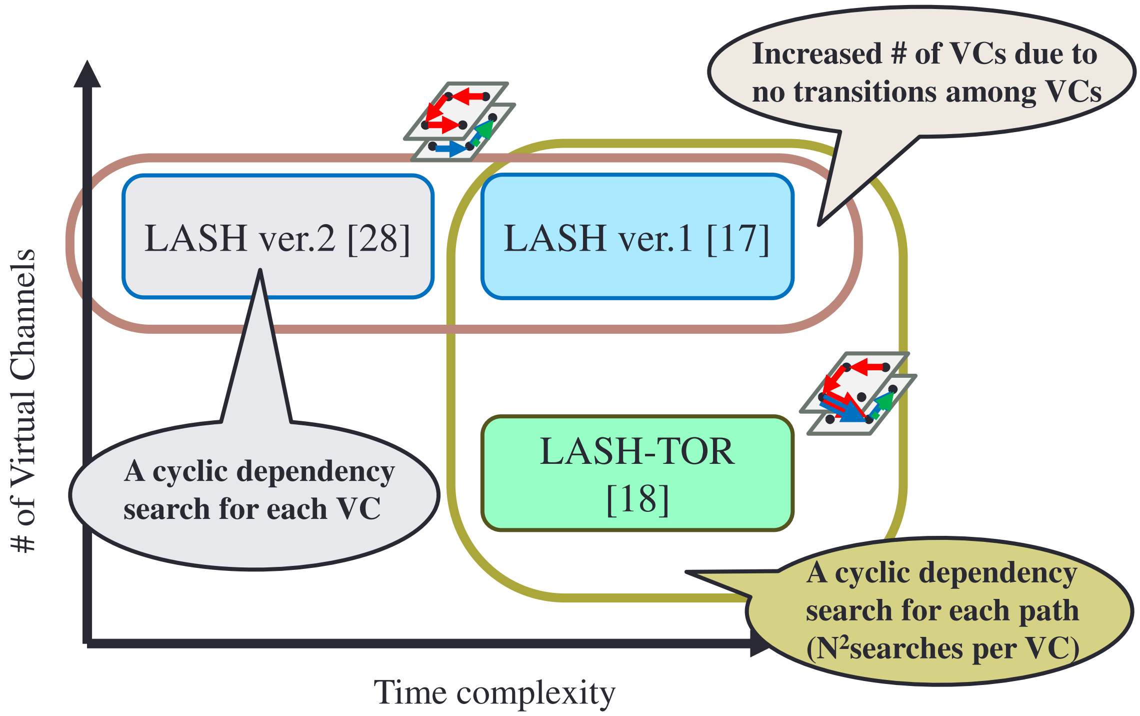

Increased # of VCs due to no transitions among VCs
LASH ver.2 [28]
LASH ver.1 [17]
LASH-TOR [18]
A cyclic dependency search for each VC
A cyclic dependency search for each path (N²searches per VC)
# of Virtual Channels
Time complexity

Figure 3.4: Qualitative comparison of deadlock-free methodologies.

of which is constructed with one of the VCs. The difference between them is whether to permit transitions among VLs for each path. In LASH routing, paths' transitions never occur, and each path must be assigned to one of the VLs, as shown in Fig. 3.3(a). It supports deadlock-free minimal paths at the cost of a relatively large number of VLs. For instance, it uses up to 9 VLs for the 256-nodes case [18]. In LASH-TOR routing, on the other hand, each path can be split into sub-paths among ordered VLs. Thus, it can achieve deadlock-freedom with only 4 VLs for the same 256-nodes case [18]. However, the time complexity can reach up to $O(|N|^3)$, which makes it quite unrealistic for large-sized networks. The trade-offs between the two routing methods are summarized in Table 3.1.

Figure 3.4 shows a qualitative comparison of three implementations of the deadlock-free methodologies, the conventional implementation of LASH denoted as *LASH ver.1* [17], the recent implementation of LASH denoted as *LASH ver.2* [29], and *LASH-TOR* [18]. In LASH ver.1 [17] and LASH-TOR [18], at least a cyclic dependency check must be done for a path. Since a cyclic dependency search has a time complexity of $O(|C| + |E|)$, the minimum time complexity per VL becomes approximately $O(|N|^3)$. In LASH ver.2 [29], on the other hand, only a cyclic dependency check per VL is needed. This can be done by initially adding all paths to a CDG and checking the cyclic dependencies for all edges. This solution can reduce the time complexity of search to $O(|N|^2)$.

To avoid cyclic channel dependencies in each VL, the ordered channels [30] are introduced that determine the following theorem.

Theorem 1 *A set of paths within a network is deadlock-free if, and only if, there exists a channels'* *ordering such that each path uses the channels in a decreasing order.*

This can be proved by a topological sort for a Channel Dependency Graph (CDG) induced by a set of paths [31].

Chapter 4

LOREN (Layout-Oriented Routing with Entries for Neighbors) for Practical Low-latency Interconnection Networks

In this chapter, the attention is focused on the locality of connections in the layout-conscious topologies introduced in Sec. 2.2 to explore routing algorithm for these topologies. These topologies have the following two characteristics. Firstly, randomly connected links between nodes can reduce the average number of hops between nodes. The achieved number of hops is comparable to that of completely randomized topologies. Secondly, the restriction of link lengths can reduce the number of hops between nodes that are close to each other on the floor. In other words, unlike completely irregular networks, they exhibit both small-world phenomena and local connections with randomly connected links whose lengths are limited.

In this work, a scalable routing method for layout-conscious random topologies LOREN (Layout-Oriented Routing with Entries for Neighbors) is proposed, which exploits irregularity and locality in these topologies to achieve both the small number of hops between nodes and small routing table sizes required. Unlike compact routing methods, LOREN utilizes local information for each node to achieve distributed routing policy for the avoidance of traffic concentration on some nodes (e.g. landmark nodes in compact routing methods).

Moreover, since the range of nodes which can be connected with direct links is limited in the layout-conscious random topologies, LOREN can much reduce the number of entries with the local information. If the number of entries becomes tens, we can use a high speed CAM (Content Addressable Memory) consisting of a set of registers and comparators easily. The routing tables contain both physically and topologically nearby neighbor nodes to ensure livelock-freedom and a small number of hops between nodes. The improved performance and the possibility of implementation on larger system sizes are evaluated and compared to those by Cowen's and Thorup's compact routing methods described in Sec. 3.1.

Table 4.1: Definition of parameters.

Parameter	Definition
n	Lengths of x- and y-coordinates
$e(i, j)$	Edge between nodes i and j
(x_i, y_i)	Coordinates of node i; $i = x_i + y_i \cdot n$
$\mathrm{md}(i, j)$	Manhattan Distance between node i and j; $\mathrm{md}(i, j) = \lvert x_i - x_j \rvert + \lvert y_i - y_j \rvert$
$h(i, j)$	Minimal # of hops between nodes i and j
N	Set of nodes; $\lvert N \rvert = n^2$
E	Set of edges
t_{max}	Maximum # of table entries for each node

4.1 Problem Definition

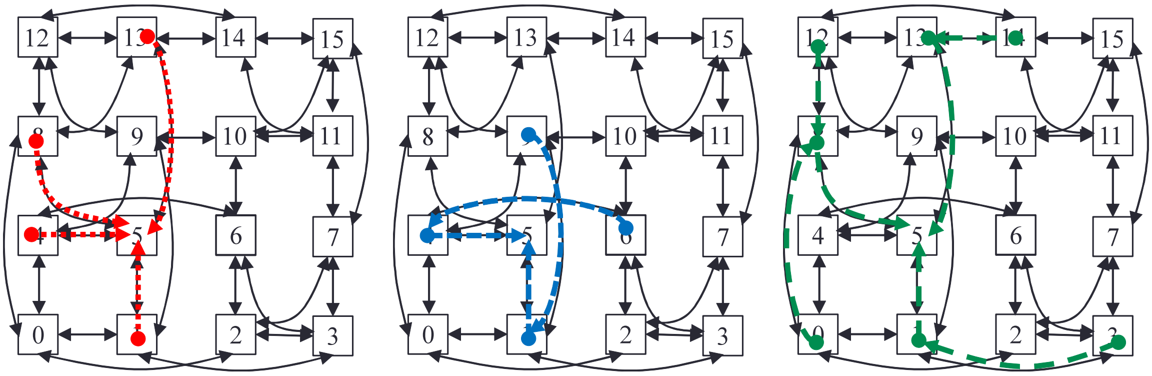
This chapter shows definitions and assumptions for the proposed method LOREN.

4.1.1 Parameters for Graphs

In this work, cabinets each of which contains a Top-of-Rack (ToR) switch and multiple computational hosts are assumed to be placed on a 2-dimensional grid. Moreover, inter-switch networks are modeled as undirected topologies, in which nodes and edges denote switches and links between switches, respectively. Topologies are developed on the $n \times n$ 2-dimensional coordinate, and the nodes are arranged on the lattice positions. Namely, the number of nodes $|N|$ and a length of one coordinate n satisfy the equality $|N| = n^2$. A length of an edge between nodes i and j, $e(i, j)$, is equal to the Manhattan Distance between the two nodes, $\mathrm{md}(i, j)$. In other words, the length is equal to $|x_i - x_j| + |y_i - y_j|$. In this work, diagonal links between nodes are not used because they degrade the overall performance with longer cable lengths [32]. Additionally, the minimal number of hops between nodes i and j is denoted with $h(i, j)$.

4.1.2 Table Entries

Each router in a network stores routing information as a routing table, which contains a set of table entries. An entry is defined as a set of two nodes and represented as a notation of $<v_{\mathrm{dst}}, v_{\mathrm{next}}>$. In this notation, v_{dst} indicates a destination node and v_{next} denotes a node of the next hop along the shortest path towards v_{dst}. The maximum number of table entries that a routing table in each node can store is defined as t_{max}. The parameters used in this work are summarized in Tab. 4.1. A goal of this work is to construct a set of routing tables with the limitation of t_{max} so that it induces an increase in the average number of hops between nodes as small as possible compared with that achieved by the shortest path routing.



(a) Shortest paths for adjacent nodes. (b) Shortest paths for neighbor nodes. (c) Shortest paths for distant nodes.

Figure 4.1: Example of shortest paths for a destination node #5 induced with table entries ($t_{\max} = 7$).

4.2 Algorithm for generating Routing Table Entries in LOREN

In this section, an algorithm in LOREN is proposed to construct routing information that is included in each node. With the routing tables generated by the proposed algorithm shown in Alg. 1, three kinds of shortest paths are constructed as described in the following Sec. 4.2.1.

4.2.1 Construction of Table Entries for Shortest Paths

In this method, the reachability of packets is supported by entries for topologically and physically neighboring destination nodes as shown in Sec. 4.2.1.1 and 4.2.1.2. The number of these entries can be reduced for the layout-conscious random topologies because of their locality. Moreover, the reduced number of hops is achieved by entries for distant destination nodes as shown in Sec. 4.2.1.3. Small-world phenomena exhibited in the topologies can reduce the number of hops with a small number of these entries. Consequently, the proposed method LOREN can reduce the number of table entries required as well as the number of hops for the layout-conscious random topologies.

4.2.1.1 Shortest Paths between Adjacent Nodes

Routing information is provided between two nodes connected with an edge in a graph. That is, for all edges $e(i, j) \in E$, an entry $<j, j>$ is added to a node i.

4.2.1.2 Shortest Paths between Neighboring Nodes on a Coordinate

On a 2-dimensional coordinate that is defined in Sec. 4.1.1, a set of two nodes i and j is neighboring where the Manhattan Distance between them $\mathrm{md}(i, j)$ is equal to one. To achieve a shortest path between i and j, an entry of $<j, m_{k+1}>$ is added to each node $m_k \in \{m_0, ..., m_{|P(i,j)|-2}\}$, where the path is represented as the following consecutive nodes,

$$P(i, j) := \{m_0(= i), ..., m_{|P(i,j)|-1}(= j)\}.$$

Algorithm 1 Generation of routing tables

```
Input: topology $I = (N, E)$, max. # of table entries $t_{\max}$
Output: routing table $T$
  /* (1) Entries for nodes connected with an edge */
  for all $(i, j) \in \{(i, j) | e(i, j) \in E\}$ do
      Add an entry $<j, j>$ to $T[i]$
  end for
  /* (2) Entries for neighboring nodes on the coordinate */
  for $(i, j) \in \{(i, j) | \mathrm{md}(i, j) = 1, e(i, j) \notin E\}$ do
      Calculate path $P(i, j) = \{m_0(= i), ..., m_{|P(i,j)|-1}(= j)\}$
      for $0 \le k \le |P(i, j)| - 3$ do
          Add an entry $<j, m_{k+1}>$ to $T[m_k]$
      end for
  end for
  /* (3) Entries for distant nodes */
  Queue $Q \leftarrow \{\}$
  for all $(i, j) \in \{(i, j) | e(i, j) \in E\}$ do
      Add a pair $(i, j)$ to tail of $Q$
  end for
  for all $j \in N$ do
      Create a spanning tree for dst $j$, $G_j$
  end for
  while $Q \neq \emptyset$ do
      pop $(u, v)$ from head of $Q$
      $w_{\mathrm{succ}} \leftarrow$ a successor of $u$ in $G_v$
      $W_{\mathrm{preds}} \leftarrow$ predecessors of $u$ in $G_v$
      if $<v, w_{\mathrm{succ}}> \in T[u]$ then
          for all $w_{\mathrm{pred}} \in W_{\mathrm{preds}}$ do
              add $(w_{\mathrm{pred}}, v)$ to tail of $Q$
          end for
      else if $|T[u]| < t_{\max}$ then
          add $<v, w_{\mathrm{succ}}>$ to $T[u]$
          for all $w_{\mathrm{pred}} \in W_{\mathrm{preds}}$ do
              add $(w_{\mathrm{pred}}, v)$ to tail of $Q$
          end for
      end if
  end while
```

Note that $<j, j>$ is already added to a node $m_{|P_{i,j}|-2}$ with the addition of entries for adjacent nodes as shown in Sec. 4.2.1.1; therefore, the entry is not added again. A packet whose destination is a node j injected to a node $i(= m_0)$ can be forwarded along this shortest path with these routing table entries. It is also notable that these entries are also utilized for the shortest path from the intermediate node $m_k \in \{m_0, ..., m_{|P(i,j)|-2}\} \setminus \{m_0\}$ towards the destination node j.

4.2.1.3 Shortest Paths between Distant Nodes

After generation of routing table entries for the two kinds of shortest paths mentioned in Sec. 4.2.1.1 and 4.2.1.2, there might be some empty routing entries in each node. These entries are utilized for some of the other shortest paths between nodes separated with multiple hops from each other.

For effective utilization of remaining empty table entries, it is desirable to give higher priority to addition of the entries for paths with small numbers of hops than to addition of those for paths with large numbers of hops. This is achieved by the following ways. Initially spanning trees are generated for all destination nodes, which are represented as $\{G_j | j \in N\}$, each of which contains shortest paths for the destination node j. After this generation, the breadth first search is applied simultaneously to all trees. In this search, when a node u in a tree G_v is visited, an entry $<v, w_{\mathrm{succ}}>$ is added to the node u if possible, where w_{succ} represents a successor node of u in the tree G_v. If the entry is successfully added to u or u already has the entry $<v, w_{\mathrm{succ}}>$, the search is continued for predecessors of u in G_v, W_{preds}. Otherwise, the search for the predecessors is discontinued. In the same way as described in Sec. 4.2.1.2, a packet whose destination is v in a node that contains an entry for v can be forwarded along the shortest path.

As described in Alg. 1, the breadth first search for all spanning trees $\{G_j | j \in N\}$ begins with initialization of the queue Q. Namely, each pair of an adjacent node i and a root node j is added to Q for all nodes $j \in N$ to process the search sequentially for all trees. The order of the pairs added to Q may cause biased traffic load. In the implementation, the pairs are added in an increasing order of the root node j, breaking ties with the adjacent node i. The impact on load balancing is evaluated and compared in Sec. 4.4.4.

4.2.2 Example of Table Entries for Destination Node

Figure 4.1 shows shortest paths for a destination node #5 induced by generated routing tables. In this example, the upper bound of a number of table entries is set to $t_{\mathrm{max}} = 7$. A starting point of a dashed arrow that is depicted with filled circle represents the node which has an entry of $<5, v_{\mathrm{end}}>$, where v_{end} is denoted by an end point of the dashed arrow. For instance, a node #0 has an entry $<5, 8>$. The dashed arrows in Fig. 4.1(a), 4.1(b), and 4.1(c) show paths for adjacent nodes, neighboring nodes, and distant nodes as described in Sec. 4.2.1.1, 4.2.1.2, and 4.2.1.3, respectively. As shown in these figures, the adjacent nodes of a node #5 are intermediate nodes for the other shortest paths to a node #5 that are described in Sec. 4.2.1.2 and 4.2.1.3. It is notable that the new entries that would be duplicated with existing entries for the adjacent nodes are not added for these shortest paths.

4.3 Routing Algorithm in LOREN

In the proposed method LOREN, after generating routing table entries as described in Sec. 4.2.1, entries in each node u are sorted in an increasing order in the number of hops between u and v_{dst}, $h(u, v_{\mathrm{dst}})$.

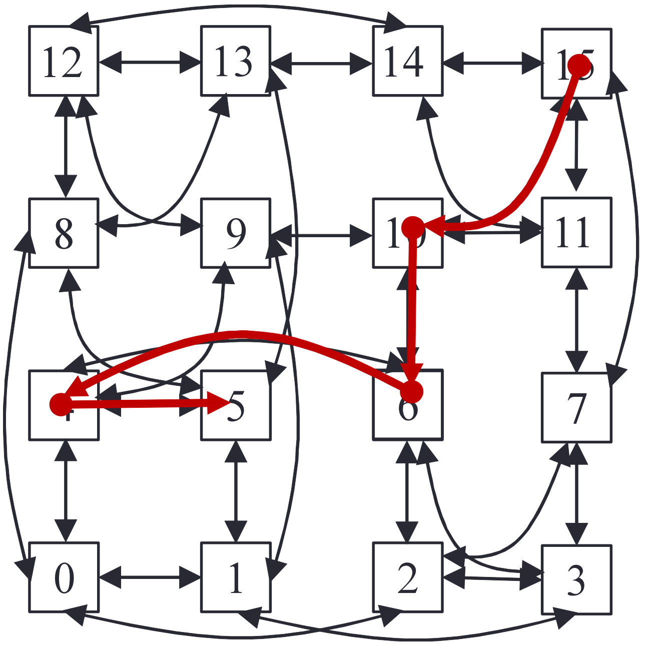
A node of the next hop for a packet in a node u whose destination node is v is determined with the following procedure. Among routing table entries in the node u, the entry $<v_{\mathrm{dst}}, v_{\mathrm{next}}>$ that minimizes the Manhattan Distance between nodes v_{dst} and v, $\mathrm{md}(v_{\mathrm{dst}}, v)$, is referred. If there are some ties, they are broken in the order as described above. This breaking of ties is achieved by referring

	T_{15}	
	Dst	Next
	7	7
	10	10
	11	11
	14	14
	2	7
	3	7
✓	**6**	*10*

(a) Routing table in node #15.

	T_{10}	
	Dst	Next
✓	**6**	*6*
	9	9
	11	11
	15	15
	1	9
	2	6
	14	11

(b) Routing table in node #10.

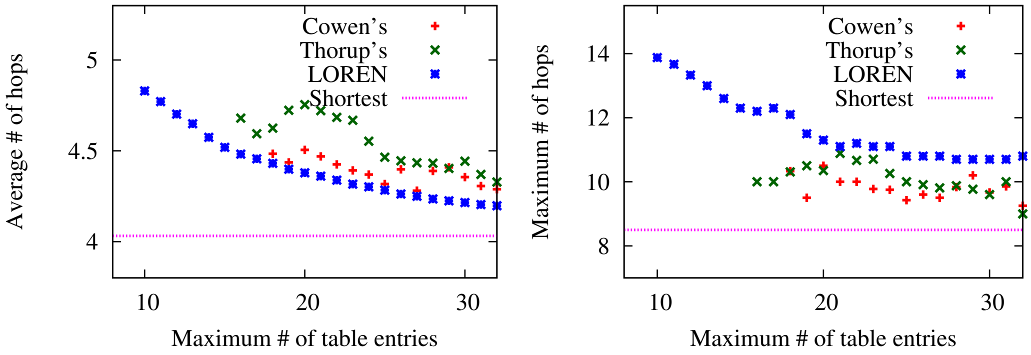


(c) Established path from node #15 to #5.

Figure 4.2: Example of routing from node #15 to #5 ($t_{\max} = 7$).

the forefront entry among the entries in a routing table of the node u, in which entries are sorted as mentioned before.

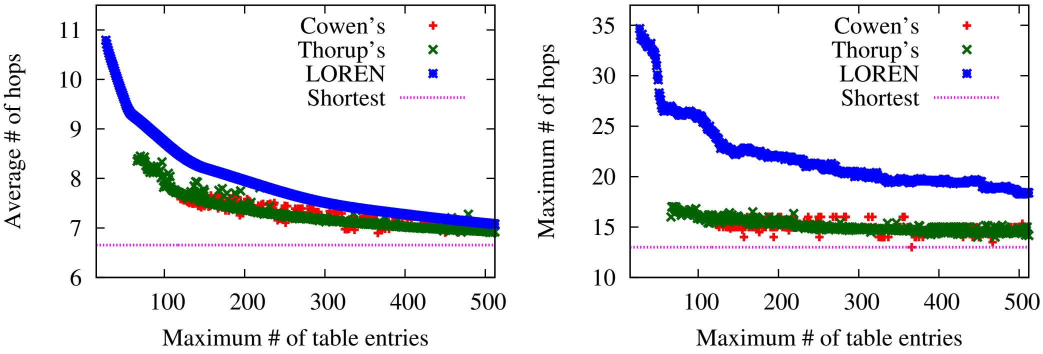
An example of routing packets is shown in Fig. 4.2. A given topology and generated table entries are the same as the example in Fig. 4.1. Let a packet be routed from a node #15 to a #5. A routing table that a node #15 has is shown in Fig. 4.2(a). For all entries in this table, the Manhattan Distance between a destination node of each entry and the destination of the packet, a node #5, is calculated. Afterwards, an entry <6, 10> is selected from these entries and referred to forward the packet, which has the smallest Manhattan Distance between nodes #6 and #5, $\mathrm{md}(6, 5) = 1$. The packet is thus forwarded to a node #10. A routing table in a node #10 is shown in Fig. 4.2(b). In the same way, an entry <6, 6> is selected that has the smallest Manhattan Distance. Note that an entry <9, 9> also



(a) average.

(b) maximum.

Figure 4.3: Maximum number of entries $t_{\max}$ and achieved path length (64 nodes).



(a) average.

(b) maximum.

Figure 4.4: Maximum number of entries $t_{\max}$ and achieved path length (1,024 nodes).

has the smallest Manhattan Distance. In this case, an entry <6, 6> is selected based on the order of entries in the table. After the packet is forwarded to the node #6, there are routing table entries for a shortest path between nodes #6 and #5, as shown in Fig. 4.1(b). The packet is then forwarded along this path.

The livelock-freedom in LOREN is proved as follows.

Proof 1 *Let $<\mathrm{tg}_v(u), u'>$ be referred in a node u to forward a packet whose destination is a node v. An order $>_v$ is defined as follows; $u >_v u'$ if one of the following conditions is satisfied.*

1. $\mathrm{md}(\mathrm{tg}_v(u), v) > \mathrm{md}(\mathrm{tg}_v(u'), v)$

2. $\mathrm{md}(\mathrm{tg}_v(u), v) = \mathrm{md}(\mathrm{tg}_v(u'), v)$ *and* $h(u, \mathrm{tg}_v(u)) > h(u', \mathrm{tg}_v(u'))$

Let me consider a packet in u for a destination v that is forwarded to u', where $v \neq u'$ is satisfied. If $h(u, \mathrm{tg}_v(u)) > 1$, u' has an entry such that $\mathrm{tg}_v(u) = \mathrm{tg}_v(u')$ and $h(u', \mathrm{tg}_v(u')) = h(u, \mathrm{tg}_v(u)) - 1$ are satisfied. This entry induces a path towards $\mathrm{tg}_v(u)$. Otherwise, u' has an entry such that $\mathrm{md}(\mathrm{tg}(u'), v) = \mathrm{md}(\mathrm{tg}(u), v) - 1$ is satisfied, which induces a path between neighboring nodes as mentioned in Sec. 4.2.1.2. In both cases, the order $u >_v u'$ is satisfied.

By forwarding the packet repeatedly, the order of a node u' strictly reduces that the packet is forwarded to. This leads to the fact that the packet can finally reach a node u' such that $\mathrm{md}(\mathrm{tg}_v(u'), v) = 0$ *and* $h(u', \mathrm{tg}_v(u')) = 1$ *are satisfied. This is because of* $\mathrm{md}(\mathrm{tg}_v(u'), v) \geq 0$ *and* $h(u', \mathrm{tg}_v(u')) \geq 1$. *In this case, u' is adjacent to v; therefore, the packet is immediately forwarded to v with an entry for adjacent nodes as described in Sec. 4.2.1.1.* □

4.4 Evaluations

In this section, the following three methods are compared.

- LOREN: the proposed routing method.
- Cowen's method [27]: a conventional compact routing method introduced in Sec. 3.1.
- Thorup's method [28]: an advanced compact routing method from Cowen's method, which reduces a local routing table size by up to $O(|N|^{1/2} \log^{3/2} |N|)$ with optimization of selection of "landmark" nodes.

Routing methods are applied to the layout-conscious random topologies that are defined in Sec. 2.2.

4.4.1 Number of Table Entries and Path Length

In this evaluation, the impact of the number of table entries to the average and maximum path lengths is analyzed. The limitation of maximum table entries $t_{\max}$ is varied to evaluate the number of path hops. In Cowen's and Thorup's methods, the number of entries for each node depends on the input parameter $0 < \alpha < 1$. Consequently, this value α also influences the achieved path lengths. In this section, all possible values of $t_{\max}$ and the corresponding maximum and average path lengths are evaluated.

Figure 4.3 and 4.4 show the results for 64- and 1,024-node layout-conscious random topologies, respectively. These topologies are generated as regular graphs, in which all nodes have the same degree. The degree of each node is set to four for both networks. Moreover, the maximum link lengths are set to two and eight for 64- and 1,024-node networks, respectively. These topologies are also evaluated with graph analysis for localized traffics in Sec. 4.4.3 and with a network simulator in Chap. 6. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown.

10 different layout-conscious random topologies are generated and used for this evaluation. Evaluated values in the figures represent the average from different topologies. In these figures, legends of the two conventional methods and the proposed method are represented as "Cowen's," "Thorup's," and "LOREN", respectively. These legends are used in the subsequent evaluations. Additionally, in this evaluation, the minimal path length is presented with a legend of "Shortest."

For 64 nodes, LOREN achieves the shorter average path lengths than Cowen's and Thorup's compact routing algorithms. Figure 4.3(a) shows that it can reduce the average number of hops by up to 7.9 % compared to Thorup's method. This is because completely or nearly shortest paths in the layout-conscious networks are achieved with localized information that LOREN provides. On the other hand, roundabout paths via "landmarks" are taken in most source-and-destination pairs with Cowen's and Thorup's compact routing methods.

As described in Sec. 3.1, Cowen's and Thorup's methods keep the maximum rate of increase in path lengths less than three, which is not supported by LOREN. Compared with Cowen's method, LOREN increases the maximum number of hops by 4.9 % and by 19.3 % at minimum for 64 and 1,024 nodes, respectively. For 1,024 nodes, there are some cases that the maximum path lengths achieved with Cowen's method are equal to the minimal path length. On the other hand, LOREN increases the maximum path length by at least 40.8 % compared with minimal routing. By contrast, LOREN suppresses the minimum increase of the average number of path hops by 0.4 % compared with Cowen's algorithm, and by 6.4 % compared with minimal routing, as shown in Fig. 4.4(a). Moreover, LOREN reduces the average number of path hops by up to 2.0 % compared to Thorup's method. These results are derived from small-world phenomena in the layout-conscious random topologies that can be utilized with its distributed routing manner.

In this evaluation, the smallest numbers of $t_{\max}$ that Cowen's algorithm can take are 18 and 116 for 64 and 1,024 nodes, respectively. Moreover, modification of the parameter α can also achieve 32 and 502 of $t_{\max}$ for 64 and 1,024 nodes, respectively, which are nearest to $\frac{|N|}{2}$. Therefore, as the maximum numbers of table entries for the evaluation in Sec. 4.4.3, Sec. 4.4.4, and Chap. 6, $t_{\max} = 18, 32$ for 64 nodes and $t_{\max} = 116, 502$ for 1,024 nodes are adopted.

4.4.2 Minimal Table Entries and Table Size Required for each Algorithm

As shown in Fig. 4.3 and Fig. 4.4, the minimum number of $t_{\max}$ that LOREN can take is smaller than that for the compact routing methods. Hence it is notable that LOREN improves the adaptation to various numbers of table entries given. In this section, this flexibility is evaluated and compared with those of Cowen's and Thorup's algorithms in detail. Namely, the minimal number of table entries and the resulted table size required for each algorithm are evaluated.

4.4.2.1 Minimal Table Entries

As described in Sec. 4.4.1, Cowen's and Thorup's compact routing algorithms vary the maximum number of table entries $t_{\max}$ with the parameter α. In this section, the smallest values of $t_{\max}$ are evaluated that the algorithms can take for given topologies. The LOREN algorithm constructs three kinds of shortest paths mentioned in Sec. 4.2.1. Among them, those between distant nodes which are defined in Sec. 4.2.1.3 are constructed using redundant empty entries; therefore, they are optional in the algorithm. By contrast, those between adjacent nodes and between neighboring nodes, which are described in Sec. 4.2.1.1 and 4.2.1.2, respectively, are indispensable for livelock-freedom in the algorithm. Accordingly, the numbers of table entries that are required for establishing these two shortest paths are evaluated.

In this evaluation, the maximum link length and the maximum degree of each node are set to four and four, respectively. Due to their time complexities to generate table entries, the number of nodes is varied among 64 to 4,096 for Cowen's and Thorup's algorithms and among 64 to 1,048,576 for LOREN. For each network size, 1,000 topologies are generated from different seeds and the average and standard deviation are evaluated.

Figure 4.5 shows that the two compact routing methods remarkably increase the required number of table entries for large system sizes. Due to the increased simulation time, the evaluation results for the case of more than 4,096 nodes by Cowen's and Thorup's compact routing methods cannot be shown. The average values for a 4,096-node layout-conscious random topology are 257 in Cowen's method and 150 in Thorup's method. These results arise from the increased numbers of "landmarks" and the numbers of the next hops that each node has to contain. On the other hand, LOREN achieves little increase in numbers of table entries for large network sizes. This is attributed to the same limitation of link lengths for every number of nodes, which can keep the number of hops between neighboring nodes almost constant. For 4,096 nodes, the number of required table entries is reduced by 91 % compared to that with Cowen's method. it is also reduced by 85 % compared to that with Thorup's method. Moreover, the increase from 4,096 nodes to 1,048,576 in LOREN is limited to 23 %.

4.4.2.2 Minimal Routing Table Size

The resulted routing table sizes of each node for Cowen's, Thorup's, and LOREN algorithms can be calculated from the results in Sec. 4.4.2.1. In addition, the routing table size required for minimal routing is calculated, which is shown with a legend of "Shortest." In this routing, it is assumed that table entries for all the other nodes are required in each node.

The results are shown in Fig. 4.6. Due to the increased simulation time, the evaluation results for the case of more than 4,096 nodes by Cowen's and Thorup's compact routing methods cannot be shown. The rate of increase with LOREN algorithm for a larger number of nodes is the smallest among all four algorithms. Since minimal routing requires the table size of $|N| \log |N|$, it needs the size of 20 M bits for 1M nodes. On the other hand, LOREN can achieve the size of 1,140 bits for the same system size. Hence, LOREN is the preferable way to achieve the next generation of high performance computing systems that have interconnection networks with 1M switching fabrics or more.

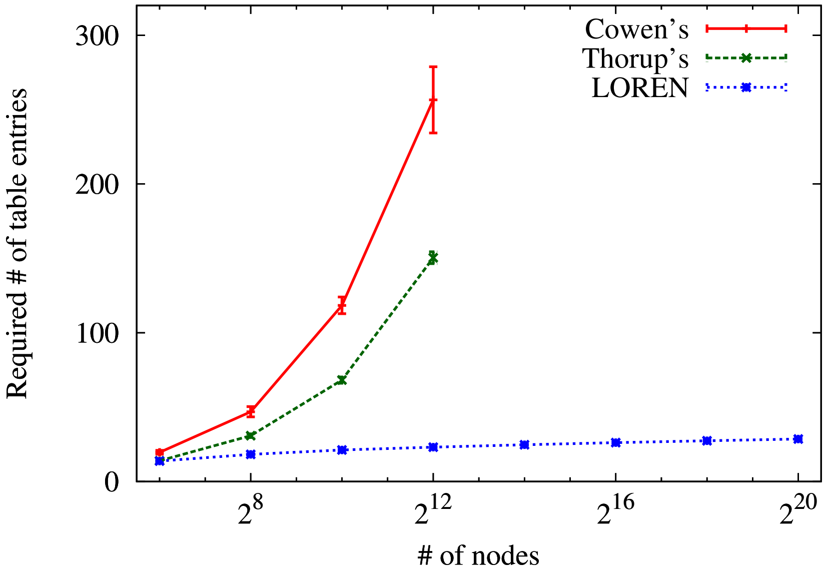


Figure 4.5: Number of table entries required for algorithms.

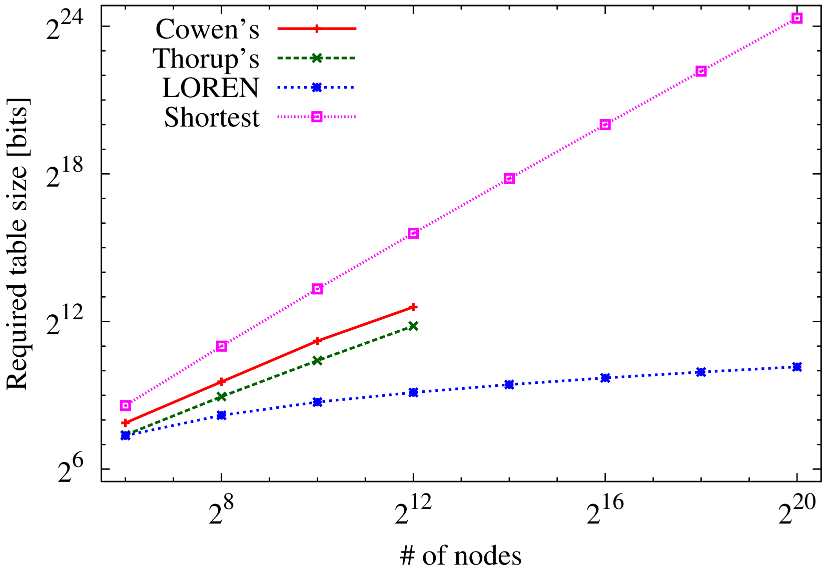


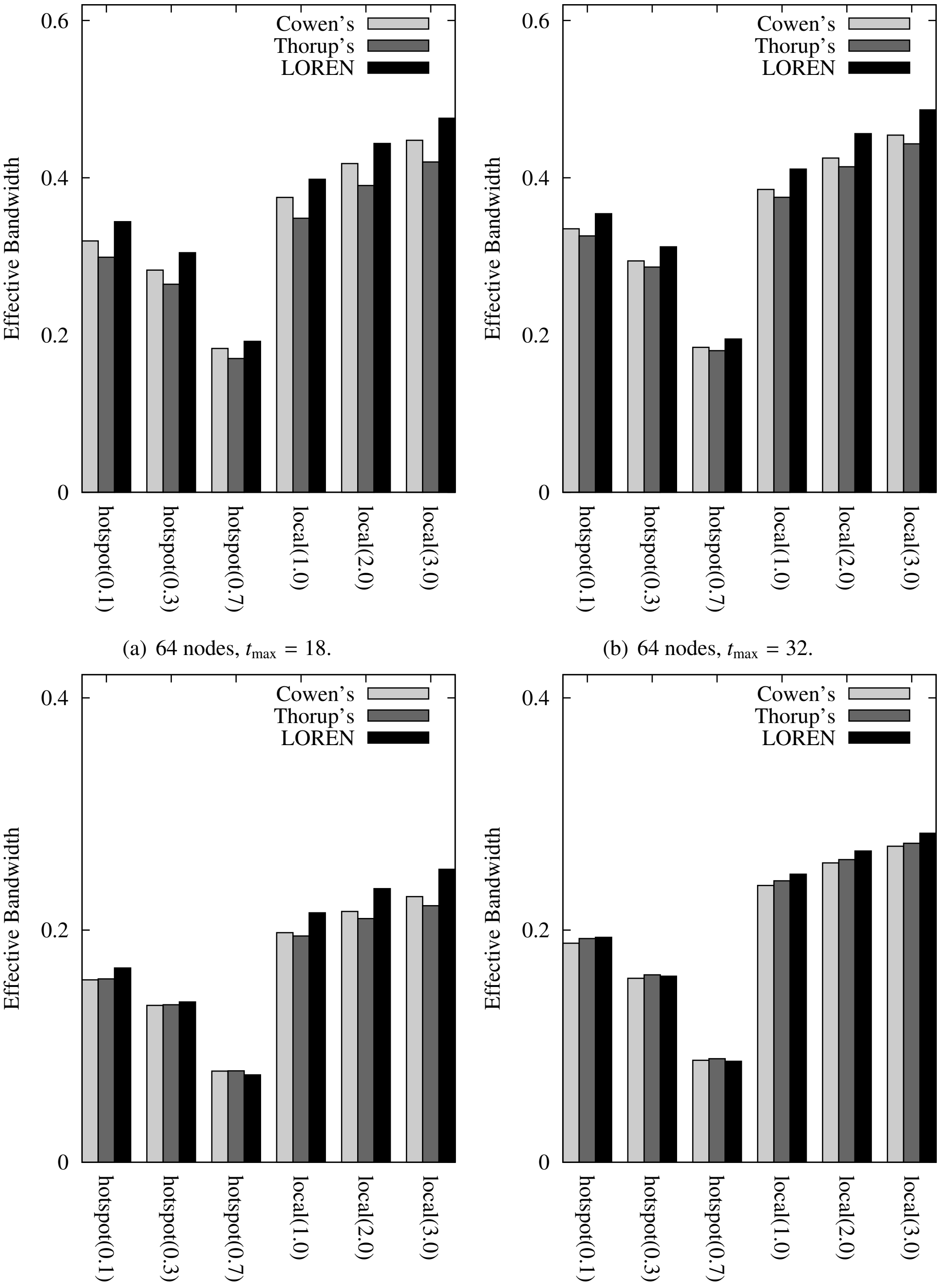
Figure 4.6: Routing table size required for algorithms.

4.4.3 Performance under Randomly Imbalanced Traffic

In this section, randomly imbalanced traffics are applied to LOREN and the two compact routing methods.

4.4.3.1 Definition of Randomly Imbalanced Traffic

In this evaluation, each node in a network has a destination node. Two imbalanced traffics, hotspot(β) and local(γ) are defined as follows.



(a) 64 nodes, $t_{\max}$ = 18.

(b) 64 nodes, $t_{\max}$ = 32.

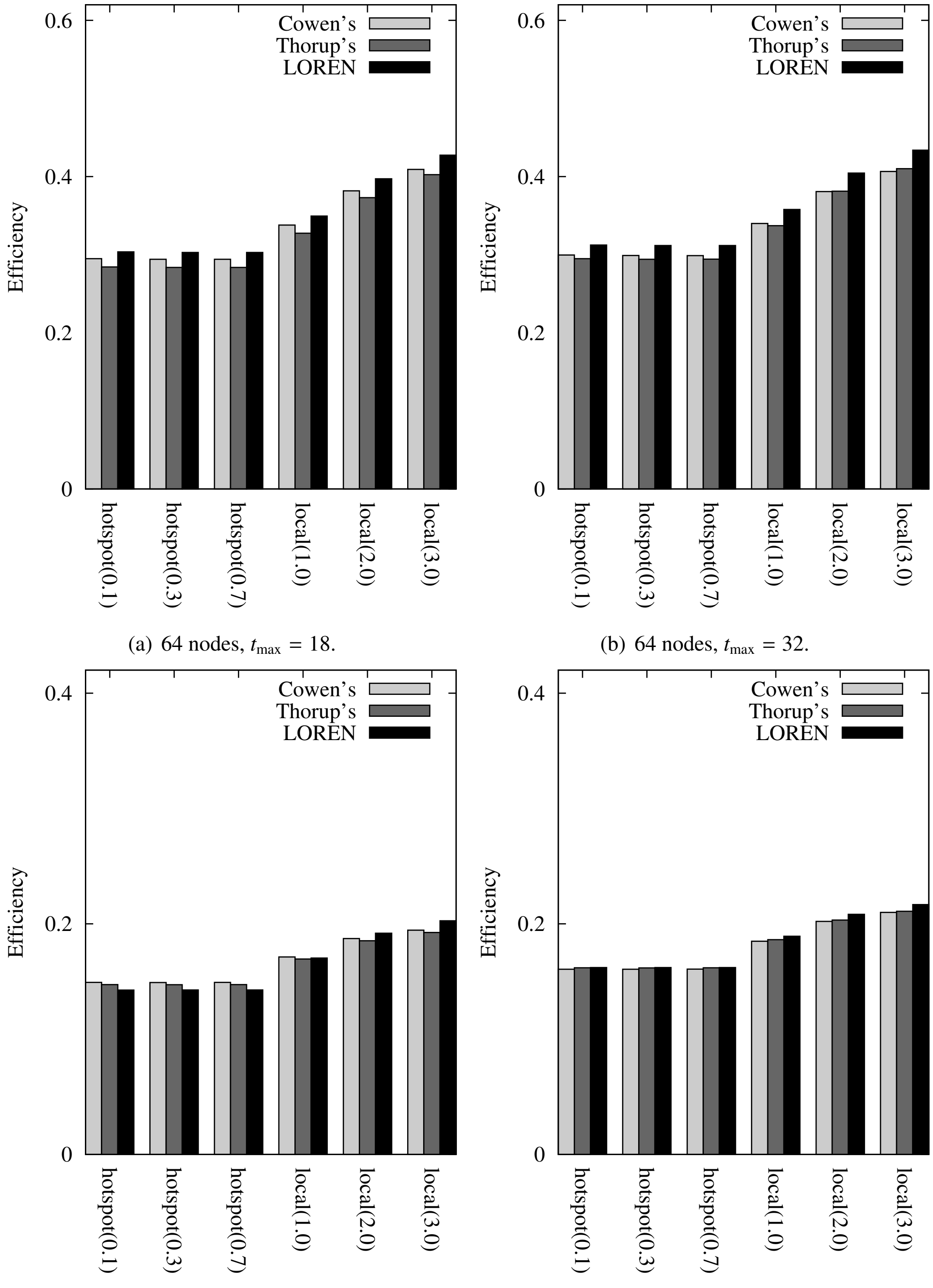
(c) 1,024 nodes, $t_{\max}$ = 116.

(d) 1,024 nodes, $t_{\max}$ = 502.

Figure 4.7: Effective bandwidth under imbalanced traffic.

- hotspot(β): The value $0 \leq \beta \leq 1$ denotes probability for each node to set a "hot-spot" node as a destination node. A destination node of each node is otherwise selected randomly from all nodes in the network with probability $1 - \beta$.

- local(γ): A destination node v for each node u is selected with the probability that is propor-



(a) 64 nodes, $t_{\max} = 18$.

(b) 64 nodes, $t_{\max} = 32$.

(c) 1,024 nodes, $t_{\max} = 116$.

(d) 1,024 nodes, $t_{\max} = 502$.

Figure 4.8: Average efficiency under imbalanced traffic.

tional to $\mathrm{md}(u, v)^{-\gamma}$, satisfying $\gamma > 0$.

Let S be a set of source-and-destination pairs in the generated traffic. A pair of a source node i and a destination node j is represented as $(i, j) \in S$. In this section, a path from i to j is represented as consecutive edges, $P'(i, j) := \{e(i, m_0), e(m_0, m_1), ..., e(m_{k'}, j)\}$, satisfying $k' = |P'(i, j)| - 2$. A set of all paths in the traffic is defined as $P' := \{P'(i, j) | (i, j) \in S\}$.

4.4.3.2 Effective Bandwidth

Let a congestion factor of an edge e, $\tau(e)$, be the number of paths in P' that include the edge e. The effective bandwidth for each traffic is defined as the following equality [33, 34].

$$\Gamma(P') = \frac{1}{|S|} \sum_{(i,j) \in S} \frac{1}{\max\{\tau(e) | e \in P'(i, j)\}}$$

In this evaluation, 10,000 random traffic patterns for each imbalanced traffic are generated, and the average effective bandwidth is calculated.

Figure 4.7 shows the results with the hotspot traffics for$\beta = 0.1, 0.3, 0.7$ and with the local traffics for $\gamma = 1.0, 2.0, 3.0$. It is shown that LOREN can achieve the better performance than Cowen's and Thorup's algorithms except in the case of the hotspot traffics for 1,024 nodes. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown. As shown in Sec. 4.4.1, LOREN increases the number of hops between nodes in these cases, which leads to a large number of edges that each path contains and thus to high degree of congestion. In contrast, LOREN effectively improves the bandwidth especially for the local traffics. This is because of the reduced number of hops between physically nearby nodes and the balanced load. When $t_{\max}$ is large, it improves the bandwidth by 10.2 % and 4.1 % for 64- and 1,024-node networks, respectively.

4.4.3.3 Average Efficiency

Efficiency for a path between nodes i and j is defined as a multiplicative inverse of the path length, $h(i, j)^{-1}$ [35]. It quantifies the characteristic of small-world phenomena developed by a routing method. The average efficiency is the average value for all paths in the traffic. In the same way as Sec. 4.4.3.2, 10,000 random patterns are generated for each traffic and the average is calculated.

Figure 4.8 shows that LOREN can improve the average efficiency by up to 6.7 % and 3.2 % for 64- and 1,024-node networks, respectively. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown. The improvement in the average efficiency is slightly smaller than that in the effective bandwidth as shown in Sec. 4.4.3.2. These results illustrate the fact that LOREN can improve the bandwidth not only by the reduced number of hops but by the balanced load achieved with the distributed routing manner. In summary, the proposed method LOREN can be efficiently applied to the practical localized or imbalanced traffic.

4.4.4 Bias of Traffic Load

In this evaluation, load on each edge, $L(e)$ is calculated as the following equality.

$$L(e) = |\{(i, j) | P'(i, j) \ni e\}|$$

An all-to-all traffic is adopted and the average and standard deviation among all edges are calculated.

Table 4.2: Load on edges under all-to-all traffic.

(a) 64 nodes, $t_{\max}$ = 18.

Method	avg.	std.
Cowen's	71.61	57.31
Thorup's	73.29	62.49
LOREN	71.82	53.22

(b) 64 nodes, $t_{\max}$ = 32.

Method	avg.	std.
Cowen's	69.09	51.95
Thorup's	69.95	52.77
LOREN	67.25	50.51

(c) 1,024 nodes, $t_{\max}$ = 116.

Method	avg.	std.
Cowen's	1966	1506
Thorup's	1990	1458
LOREN	2184	1165

(d) 1,024 nodes, $t_{\max}$ = 502.

Method	avg.	std.
Cowen's	1804	1138
Thorup	1784	1077
LOREN	1826	1056

Table 4.2 shows the results. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown. For all cases the standard deviation with LOREN is smaller than that with Cowen's and Thorup's compact routing methods, which means that LOREN can balance the traffic load compared to compact routing algorithm. This measurement is obvious for the case of 1,024 nodes and $t_{\max} = 116$. Although the average traffic load is increased by 11.1 %, the standard deviation is reduced by up to 22.6 %. Therefore, LOREN is a more scalable and preferable way to construct interconnection networks with larger numbers of nodes and smaller numbers of table entries.

4.4.5 Load Distribution

In this section, load distribution on layout-conscious random topologies is visualized to compare the proposed algorithm LOREN and a conventional compact routing method. Due to the increased simulation time, the evaluation results for the case of 1,024 nodes are shown.

Thorup's compact routing algorithm generates a *cluster* for each *landmark* node. As shown in Sec. 3.1, Thorup's algorithm sets a portion of nodes in a given topology as landmark nodes to route packets. Each node in the topology is grouped into one of the clusters. When a destination node of a packet is distant from a current source node, the packet is forwarded towards one of the landmark nodes that the destination node belongs to. Figure 4.9 shows an example of landmarks and their clusters induced by Thorup's algorithm. In this example, 18 landmarks labeled with 'L' are set among 1,024 nodes in the network. The landmarks are colored with the different colors from each other. Each landmark generates a cluster that is denoted as a set of nodes that are colored with the same color.

Figure 4.10 shows a heat-map scale of load distribution for a uniform traffic on the topology and the landmarks that are shown in Fig. 4.9. The uniform traffic is generated randomly for 1,000 times. The number of paths that pass through each node is calculated to evaluate the average value for each node. The heat-map scale is generated using the minimum and the maximum values, 2,149 and 75,544, in the entire values of all nodes. The high, middle, and low loads are displayed with red, white, and blue colored squares, respectively.

The results demonstrate that Thorup's algorithm cannot distribute the traffic load, especially on the landmark nodes. It is observed that, while the landmark nodes have the high load, the peripheral routers are low utilized. This behavior derives from the concentration of the packets from the nodes distant to the landmark nodes.

On the other hand, LOREN can avoid traffic concentration for all nodes in the topology with distributed local routing information for each node. Figure 4.11 shows a heat-map scale of load distribution induced by LOREN on the same topology as in Fig. 4.11. In this evaluation, a uniform traffic is generated randomly in the same way as Fig. 4.10. Moreover, the same maximum and minimum values are used to generate the heat-map scale. The results illustrate that LOREN can reduce the average traffic load for all nodes in the topology. Any high traffic load does not occur on certain nodes, which is seen on the landmark nodes in Thorup's algorithm as shown in Fig. 4.10. Moreover, compared to Thorup's algorithm, the load is well-flattened between the center nodes and the edge nodes.

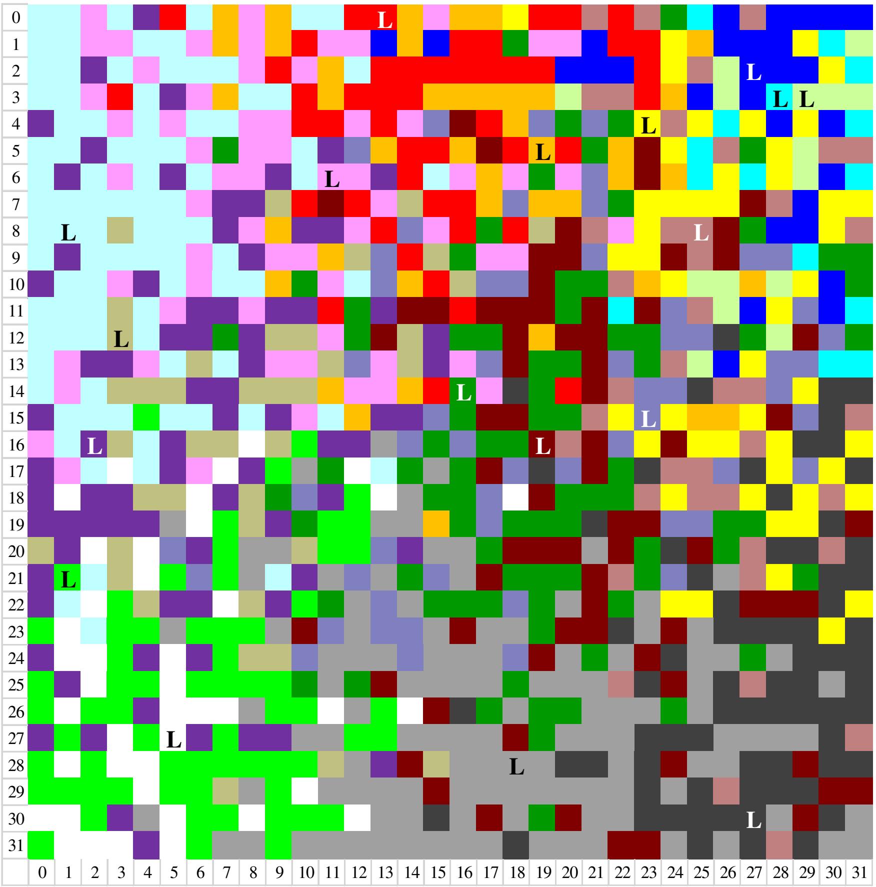


Figure 4.9: Example of landmarks and their clusters induced by Thorup's algorithm (18 landmarks and clusters).

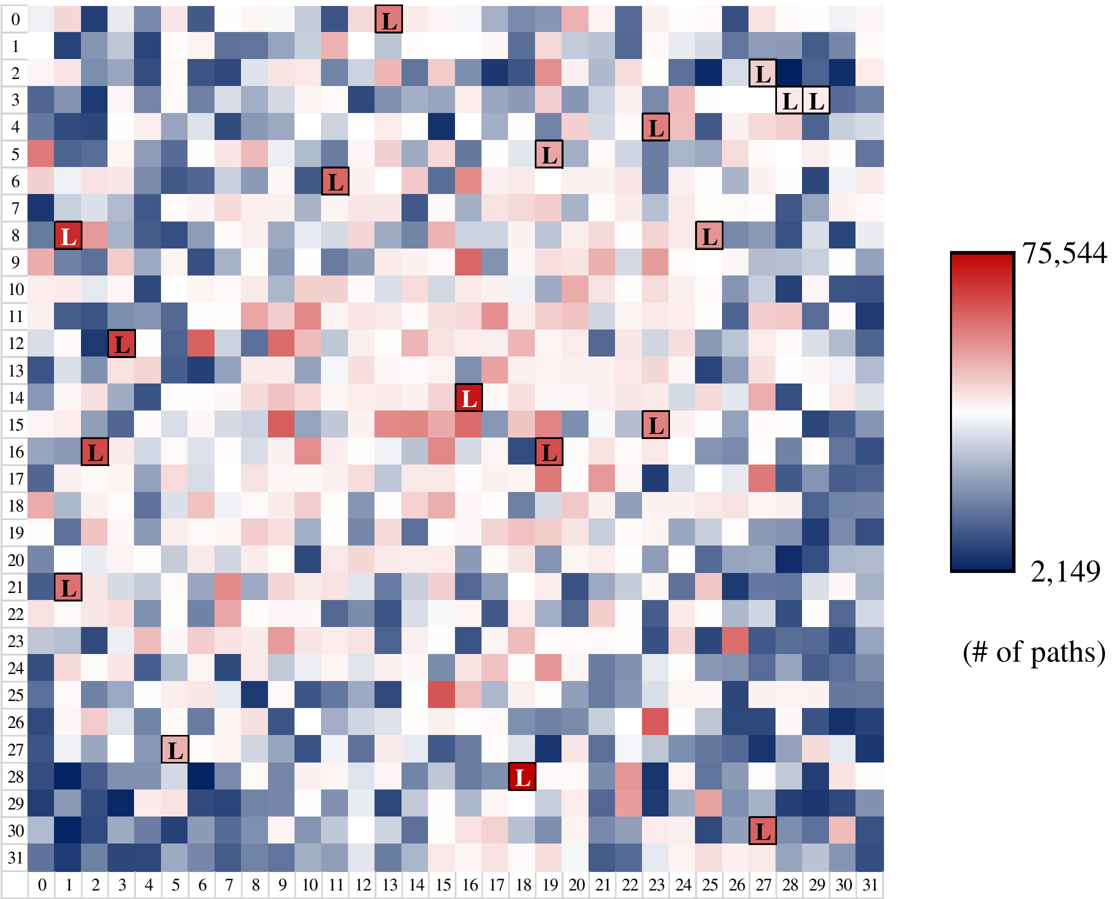


Figure 4.10: Load distribution by Thorup's algorithm (with 18 landmarks, 75 entries) across all switches in 32x32 layout-conscious random topologies (the maximum link length of 8, the degree of 4) on a heat-map scale.

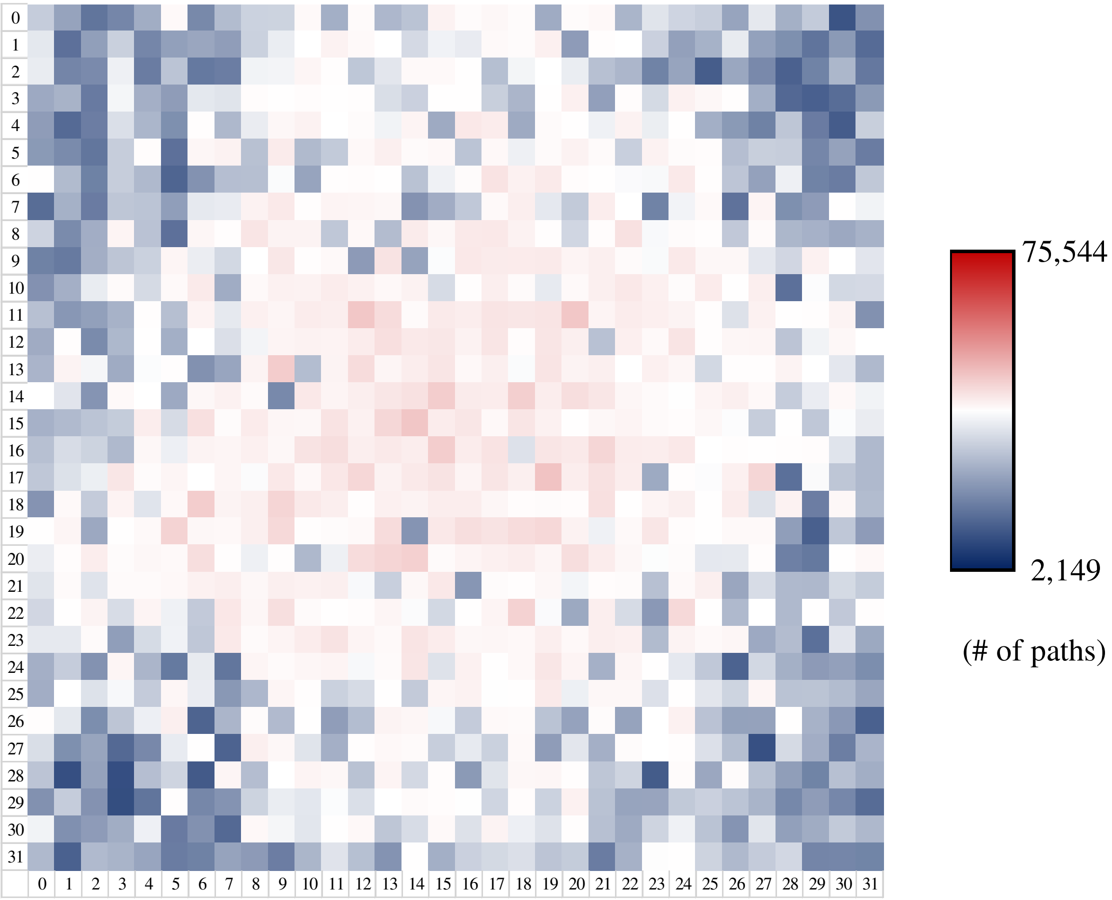


Figure 4.11: Load distribution by LOREN (with 116 entries) across all switches in 32x32 layout-conscious random topologies (the maximum link length of 8, the degree of 4) on a heat-map scale.

Chapter 5

ACRO (Assignment of Channels in Reverse Order) for Deadlock-free Routing

In this chapter, multiple Virtual Channels (VCs) for each physical channel are exploited, as used in LASH [17, 29] and LASH-TOR [18] routings described in Sec. 3.2, in order to support deadlock-freedom for arbitrary routing methods. The proposed methodology named *Assignment of Channels in Reverse Order (ACRO)* takes a given topology and the routing table, obtained from a livelock-free routing algorithm, as inputs to generate the VC assignment to paths. This approach has a small time complexity, yet with the same number of VCs when compared with conventional methods.

5.1 Problem Definition

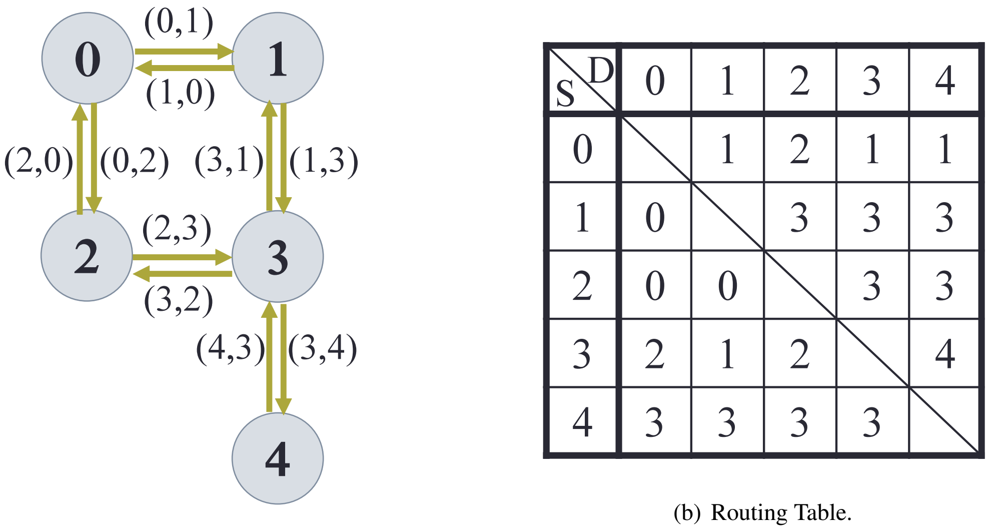
According to the general models [30], the configuration of an interconnection network is defined as follows.

Definition 1 *An interconnection network I is represented by a directed graph $I = (N, C)$, where N is a set of switches and C is a set of physical channels.*

Definition 2 *A deterministic routing function $R : N \times N \rightarrow C$ returns the physical output channel c_{out} to be taken from a node n_{i} for packets whose destination node is n_{d}.*

In the methodology, a topology and a routing table, represented in Fig. 5.1, are given as an interconnection network and a routing function, respectively. The routing table in Fig. 5.1(b) takes a current node n_{i} and a destination node n_{d} as inputs, and returns the next node n_{next}. Therefore, the output channel c_{out} in Def. 2 is determined as follows: c_{out} is $(n_{\mathrm{i}}, n_{\mathrm{next}})$ if $n_{\mathrm{i}} \neq n_{\mathrm{d}}$; otherwise, c_{out} becomes $c_{\mathrm{local},n_{\mathrm{d}}}$.

The definition of a VL L_i in a network I is the same as that adopted in LASH-TOR [18]; that is, L_i can be treated as a virtual network that is isomorphic to the original physical network I. Additionally, in this work, L_i is defined as a strictly and decreasingly ordered set. It contains sorted physical



S \ D	0	1	2	3	4
0		1	2	1	1
1	0		3	3	3
2	0	0		3	3
3	2	1	2		4
4	3	3	3	3	

(b) Routing Table.

(a) Topology.

Figure 5.1: An example of given inputs.

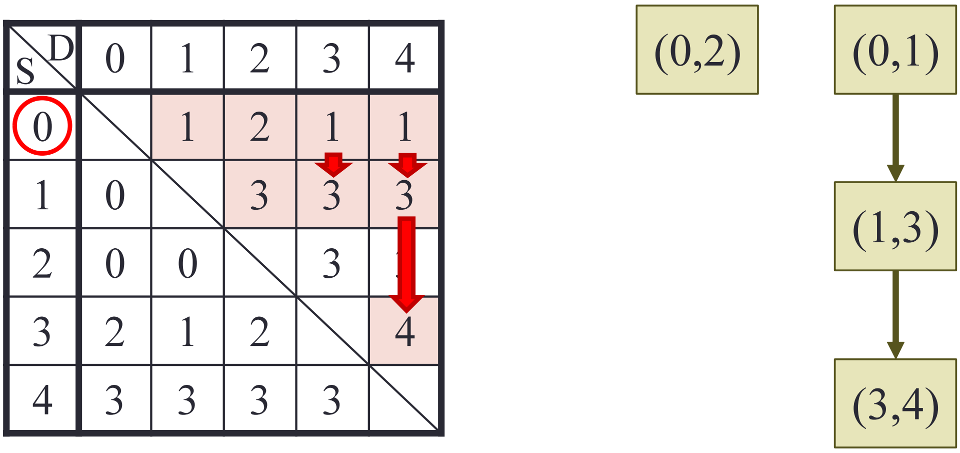
channels of I such that every path within L_i is restricted to use the corresponding VCs in a decreasing order. Namely, each VL is a sorted set of C and is denoted as $L_i := \{c_{i,|C|-1}, ..., c_{i,0}\}$.

Given the inputs of a topology and a routing table, a strictly and decreasingly ordered set of layers $L = \{L_{|L|-1}, ..., L_0\}$ is determined such that for every (source, destination) pair, the path can reach the destination by using channels in decreasing order within each L_i and transitions among layers in a decreasing order within L.

5.2 CDG (Channel Dependency Graph) Generation for Each Destination

As shown in Sec. 3.2, in the conventional implementation of LASH-TOR [18], at least a cyclic dependency check must be done for a path. Since a cyclic dependency search has a time complexity of $O(|C| + |E|)$, the minimum time complexity per VL becomes approximately $O(|N|^3)$. In the recent improvement on LASH [29], only a cyclic dependency check per VL is needed. This can be done by initially adding all paths to a CDG and checking the cyclic dependencies for all edges. Detected cycles are removed by moving a portion of paths included in each cycle to the next VL. Then, the dependency check is done again. It is iterated until no cyclic dependency is detected. This solution is called an offline-manner which can reduce the time complexity of search to $O(|N|^2)$. Here, the extension of this offline-manner for LASH-TOR, called ACRO is proposed to balance the number of VLs and the time complexity.

The details of the proposed ACRO algorithm are shown in Alg. 2. A set of paths from a set of nodes N to a destination node n_d is determined as $T'_{n_\mathrm{d}} = (N, C_{n_\mathrm{d}})$, satisfying $C_{n_\mathrm{d}} \subset C$. T'_{n_d} forms a directed tree whose root is n_d, and all edges are directed toward the direction of n_d. T'_{n_d} produces a

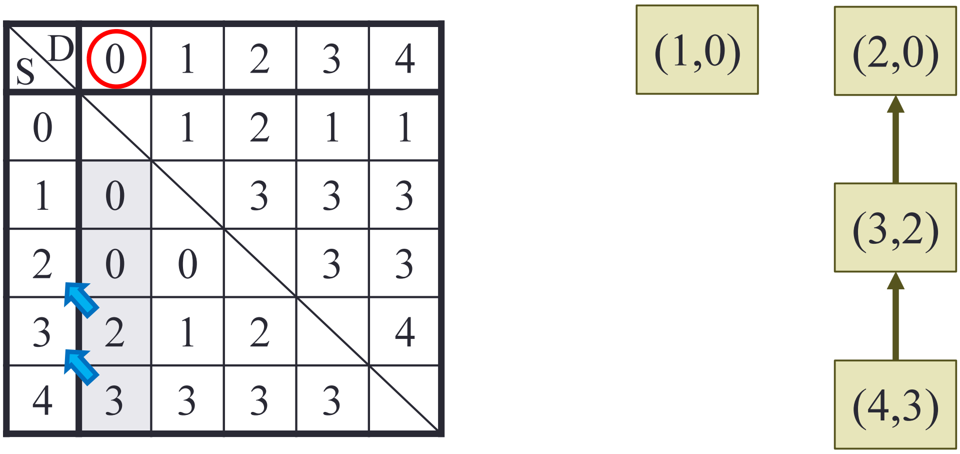


S D	0	1	2	3	4
0		1	2	1	1
1	0		3	3	3
2	0	0		3	3
3	2	1	2		4
4	3	3	3	3	

(a) Routing Table.

(b) CDG for src. #0.

Figure 5.2: Creation of CDG for src. #0.



S D	0	1	2	3	4
0		1	2	1	1
1	0		3	3	3
2	0	0		3	3
3	2	1	2		4
4	3	3	3	3	

(a) Routing Table.

(b) CDG for dst. #0.

Figure 5.3: Creation of CDG for dst. #0.

CDG for the destination n_d. Here, it is represented as $T_{n_\mathrm{d}} = (C, E_{n_\mathrm{d}})$, where E_{n_d} denotes a set of the channel dependencies. T_{n_d} is a set of directed trees, as shown in Fig. 5.3(b), and the node of the CDG is corresponding to 'channel.' To avoid any confusion, a node in a CDG is called a 'channel.'

In this work, all of the channel dependencies with all paths in a traffic are determined as a set of CDGs for the destination nodes rather than the source nodes. This choice can be explained by the fact that, and as shown in Fig. 5.2(b), if a CDG is generated for each source node, the number of referred elements in the table for each destination node is equal to the number of hops between the source and destination nodes (Fig. 5.2(a)). Therefore, the time complexity becomes $O(|N| \cdot \log |N|)$ for each source node, where $O(\log |N|)$ comes from the characteristics of irregular networks [11]. On the other hand, and as depicted in Fig. 5.3(b), if a CDG is generated for each destination node, only a column of the table is needed to be referred to (Fig. 5.3(a)). As a result, the time complexity becomes only $O(|N|)$ for each destination node.

5.3 Heuristic Approach to Reduce VLs

In this work, a heuristic approach to reduce VCs is achieved with a hash table, which improves the scalability of the algorithm.

5.3.1 Weight Values of Channel in CDG

VCs in each VL should be ordered properly to minimize the number of required VLs for deadlock-freedom. In this section, a simple heuristic approach is introduced to minimize the number of paths and the length of each path moved to the next VL.

$C_{\text{child}}(n, c)$ is defined as a set of channels which are children of the channel c in a CDG T_n. Two weight values, $h_{n,c}$ and $w_{n,c}$, are introduced for c. $h_{n,c}$ represents the "height" of c, and $w_{n,c}$ represents the number of the "deepest" leaves of c. These values are calculated by the following equations:

$$h_{n,c} = \begin{cases} 0 & (\text{if } C_{\text{child}}(n, c) = \emptyset) \\ 1 + \max H_{\text{child}}(n, c) & (\text{otherwise}) \end{cases}$$

where

$$H_{\text{child}}(n, c) := \{h_{n,c'} | c' \in C_{\text{child}}(n, c)\}$$

and

$$w_{n,c} = \begin{cases} 1 & (\text{if } C_{\text{child}}(n, c) = \emptyset) \\ \displaystyle\sum_{c' \in C'(n,c)} w_{n,c'} & (\text{otherwise}) \end{cases}$$

where

$$C'(n, c) := \{c' | c' \in C_{\text{child}}(n, c), h_{n,c'} = \max H_{\text{child}}(n, c)\}$$

5.3.2 Hash Table for Each Channel

After introducing the weight values of a channel in a given CDG, it has to be determined which channel of the cycle to be broken in order to minimize the length and the number of paths moved to the next VL. Consequently, this leads to minimize the number of VLs needed. In the proposed algorithm, a hash table for each channel c, $\mathbb{H}_c$, is introduced. It manages the number and length of the longest paths that would be moved to the next VL by the channel.

Note that the longest paths are adopted as the objectives of the heuristic, rather than all paths including shorter paths. This is because considering all paths would make the memory complexity in the algorithm quite large. Moreover, the number of dependencies to be solved is dominant to the longest paths, and shorter paths are negligible.

The values of the tables are initially calculated as follows. If a channel c has any parent in T_n, a value in $\mathbb{H}_c$ corresponding to the key $h_{n,c}$ is incremented by $w_{n,c}$. Note that this increment is not applied in the case of a root channel c because if the smallest order is set to this root channel, no path would be moved to the next VL.

5.4 Assignment of VLs to Paths in Reverse Order

To make sure whether a channel is reachable in T_n, an $|N| \times |C|$ boolean table is introduced. Each element of the table is initially set to a boolean value 'False', which is specified by $m(n, c)$ in Alg. 2. The termination condition of the algorithm is that all channels $c \in C$ are reachable in T_n for all destinations $n \in N$.

The VLs and VCs are assigned for each path in the reverse order. Namely, unlike the conventional virtually layered networks, the VLs and VCs are assigned in the order from the destination node to the source node.

After generating a new VL L_i as an empty ordered set, the order of channels is fixed in a one-by-one fashion. Among the channels whose orders are not assigned yet, a channel $u_{\min}$, which minimizes the largest key with a non-zero value in H_u, $f(u)$, is selected to append to the head of L_i. If there are some ties, they are broken by selecting one of them which minimizes the value of $\mathbb{H}_u[f(u)]$.

For each CDG T_n, if the channel $u_{\min}$ does not have a parent in T_n, the following procedures are performed. Since the packet whose destination is n can reach $u_{\min}$ in the current VL L_i, the boolean value of 'True' is assigned to $m(n, u_{\min})$. Furthermore, edges from all the children of $u_{\min}$ to $u_{\min}$ itself are removed if they exist.

The deletion of the edges denotes that the dependency between the child of $u_{\min}$ and $u_{\min}$ will be dissolved in either of the following two ways. If the order of the child is not assigned yet, it will be inevitably larger than that of $u_{\min}$. This means that the dependency is dissolved within the current L_i. Otherwise, the order of the child is surely smaller than that of $u_{\min}$, which cannot dissolve the dependency. Even after all the channel orders are assigned in L_i, the termination condition is not satisfied. This leads to the generation of a VL L_{i+1}. The dependency will be dissolved by the transition between the child in L_{i+1} and $u_{\min}$ in L_i.

After its execution, the deletion is reflected to the values of the hash tables by the decrement of $\mathbb{H}_{c'}[h_{n,c'}]$ by $w_{n,c'}$ for each child of $u_{\min}$, $c' \in C_{\text{child}}(n, u_{\min})$.

5.4.1 Deadlock-Free Routing with ACRO

This chapter shows a possible implementation and shows the proofs of the livelock- and deadlock-freedom for the proposed method ACRO.

5.4.1.1 Implementation on Switches

A possible implementation of ACRO is that each switch has a mapping table for VLs. The table holds boolean values which represent whether an order of each input channel is larger than that of each output channel.

Packets are restricted to be injected to an output channel of the first hop with a VL $L_{|L|-1}$, which has the maximum order among all VLs. In the subsequent hops, the table entries mentioned above are referred to in each switch. If the input channel has larger order than the output channel within a VL L_i that is used in the input, the same VL L_i is used in the output; otherwise, L_{i-1} is used. It is important to mention that the number of table entries for each switch is independent of the number of switches $|N|$; but, it depends on the number of input and output channels and the number of required VLs. This reduced amount of information for VL mapping enables a scalable implementation for larger system sizes.

5.4.2 Livelock- and Deadlock-Freedom with ACRO

Livelock- and deadlock-freedom supported by ACRO is proved by the following two theorems:

Theorem 2 *A path exists between an arbitrary source-and-destination pair with ACRO.*

Proof 2 *With the proposed algorithm described in Alg. 2, a path between an arbitrary source-and-destination pair is contained in a strictly ordered set of layers $L' = \{L_{|L'|-1}, ..., L_0\}$. When $|L'| > 1$ is satisfied, transitions between layers are performed in the switches given by the strictly ordered set $N' = \{n_{|L'|-2}, ..., n_0\}$. Deadlock-freedom among VLs is supported with this path division. Moreover, each subpath in each VL uses the channels in a strictly decreasing order, which supports deadlock-freedom within each VL.* □

Theorem 3 *The VC and VL assignment with ACRO and its implementation described in Sec. 5.4.1.1 can endorse a given topology and a routing table with both livelock- and deadlock-freedom.*

Proof 3 *Each VC belonging to c in a VL L_i is labeled with a two-digit identifier $(i, \mathrm{Idx}(i, c))_{|C|}$, where Idx$(i, c)$ is the order of c in L_i, and $(i, j)_{|C|} = i \cdot |C| + j$. Given these labeling to VCs, a packet in each switch with the implementation in Sec. 5.4.1.1 uses a pair of input and output channels which are labeled in strictly decreasing order. Since all packets are initially injected to the VL with the maximum order, VCs traversed by the packets always have labels equal to or larger than those in the same channel containing the established paths in Theorem 2; therefore, packets are transferred with VCs whose labels are in a strictly descending order until reaching the destination nodes.* □

5.5 Evaluations

In this section, the proposed VC assignment method is evaluated and compared with the conventional VC assignment methods proposed in LASH and LASH-TOR. As shown in Section 4.2, a topology of switches and a routing table are given as inputs. Note that the routing table takes source and destination nodes, and returns the next node.

5.5.1 Number of Required VLs

In this section, the impact of the network size and the node degree to the number of VLs is analyzed. Here, the degree is corresponding to the number of ports of a switch.

5.5.1.1 Implementation of VC Assignment Algorithm

In this evaluation, the VC assignment algorithm in LASH is implemented as follows. Let $\mathrm{G}_{\mathrm{CD},i}$ be a CDG created by a set of paths in L_i. For a given path between a source node n_s and a destination node n_d, the algorithm searches a set of VLs L to find $L_i \in L$. The path induces the channel dependencies, which could be added to $\mathrm{G}_{\mathrm{CD},i}$ without generating a cycle of channel dependencies. If L_i is found, the dependencies are added to $\mathrm{G}_{\mathrm{CD},i}$. Otherwise, a new VL and the corresponding CDG are created and the channel dependencies are added to the new CDG.

The VC assignment algorithm in LASH-TOR is implemented in a similar way as that in LASH: let G_{CD} be a set of CDGs created by sets of paths in a set of VLs L. For the same inputs as in LASH, the algorithm searches L and N to find $\{L', S\}$, satisfying $L' \subseteq L$ and $S \subset N$. The path would be split into the subpaths according to the transition node $s_j \in S$. Each subpath would induce the channel dependencies, which could be added to each $\mathrm{G}_{\mathrm{CD},i} \in \mathrm{G}_{\mathrm{CD}}$ without generating a cycle of dependencies within each $L_i \in L'$ respectively. If $\{L', S\}$ is found, the dependencies are added to G_{CD}; otherwise, a new VL and the corresponding CDG are created and added to L and G_{CD}, respectively. G_{CD} then incorporates the dependencies obtained from $\{L', S\}$ that satisfies the condition mentioned above, and is exhibited by the additional VL. The original implementation of LASH-TOR can limit the number of VLs by permitting non-minimal paths with up*/down* routing on the final VL. On the other hand, the proposed method ACRO completely supports paths induced by a given routing table. Thus, it does not use the alternative longer paths. For fairness of comparison, the number of maximum VLs are not limited in LASH-TOR; therefore, up*/down* routing is not used in the last VL during this evaluation.

5.5.1.2 Experimental Results

Surely connected regular random topologies are adopted in this evaluation, in which all of the edges are bidirectional. The number of nodes is set to $|N| = 64$ and 256. Due to the increased simulation time, the evaluation results for the case of more than 256 nodes cannot be shown. The number of degree d is varied from 4 to 12. In this evaluation, a hundred topologies are generated from different seeds for each $(|N|, d)$ pair. The corresponding routing tables take exactly one minimal path for a source-and-destination pair.

Fig. 5.4 and Fig. 5.5 show the maximum, minimum, and average numbers of required VLs for 64- and 256-node topologies, respectively. These results show that ACRO efficiently reduces the number of VLs compared with the VC assignment methodology in LASH routing. For 64-node topologies, it reduces the average and maximum numbers of required VLs by up to 37% and 50%, respectively. Furthermore, for 256-node topologies, it reduces the average and maximum numbers of required VLs by up to 60% and 63%, respectively. Another interesting result is that the heuristic used in ACRO achieves almost the same number of required VLs as the VC assignment methodology in LASH-TOR routing in which paths are assigned to VLs sequentially.

Moreover, ACRO accomplishes as small variance in the number of required VLs as LASH-TOR. In this evaluation, a difference of 2 between the maximum and the minimum numbers of required VLs is observed for LASH in the case of $(|N|, d) = (64, 3)$. On the other hand, the differences for ACRO and LASH-TOR never exceed 1.

5.5.2 Time and Memory Complexity

The original algorithm for the VC assignment in LASH [17] is accelerated by a recent implementation [29]. In this implementation, the time complexity and the memory complexity of VC assignment algorithm are as follows:

Proposition 1 *The time complexity of the algorithm for VC assignment in LASH is*

$$O(\nabla \cdot (|C| + |E|) + |N|^2)$$

while the memory complexity is

$$O(\nabla \cdot D(I) \cdot |N|^2 + \nabla \cdot (|C| + |E|)).$$

In this proposition, ∇ and $D(I)$ are a number of required VLs and the diameter of a topology, respectively. Parameters in the above proposition are to be hereinafter used.

On the other hand, the time and memory complexities of VC assignment algorithm in LASH-TOR [18] are as follows:

Proposition 2 *The time complexity of the algorithm for VC assignment in LASH-TOR is*

$$O(|N|^2 \cdot D(I) \cdot \nabla \cdot (|C| + |E|))$$

while the memory complexity is

$$O(\nabla \cdot (|C| + |E|))$$

From the proposed ACRO algorithm shown in Alg. 2, the time and memory complexities are summarized as follows. The generation of CDGs for all the destinations has a time complexity of $O(|E| \cdot |N|)$ and a memory complexity of $O(|C| \cdot |N|)$. A time complexity to calculate all weight values $h_{n,c}, w_{n,c}$ is $O(|E| \cdot |N|)$, while the memory complexity is $O(|C| \cdot |N|)$. Initialization of the values in hashes $\mathbb{H}_c$ has a time complexity of $O(|C| \cdot |N|)$ and a memory complexity of $O(|C| \cdot D(I))$. Checking whether $u_{\min}$ has any parent in T_n needs a time complexity of $O(\nabla \cdot |C| \cdot |N|)$ in total. The selection of the channel which minimizes the fitness function $f(u)$ has a time complexity of $O(|C|^2 \cdot \nabla)$ in total. Moreover, the modification of $\mathbb{H}_{c'}$ and $f(c')$ needs time complexities of $O(\nabla \cdot |C| \cdot |N|)$ and $O(\nabla \cdot |C| \cdot D(I))$ in total, respectively.

The findings mentioned above are followed by these propositions:

Proposition 3 *The time complexity of the ACRO algorithm is*

$$O(\nabla \cdot |C| \cdot (|N| + |C| + D(I)) + |N| \cdot |E|)$$

while the memory complexity is

$$O(|C| \cdot (|N| + D(I)).$$

Given that a topology is randomly generated and the degree d is quite smaller than the network size $|N|$, the proportionalities $D(I) \propto \log |N|$, $|C| \propto |N|$, and $|E| \propto |N|$ are satisfied [11]. From these proportionalities, it can be said that ACRO reduces the time complexity by a factor of $O(|N| \cdot \log |N|)$ when compared with that of the VC assignment algorithm in LASH-TOR, yet with almost the same number of required VLs.

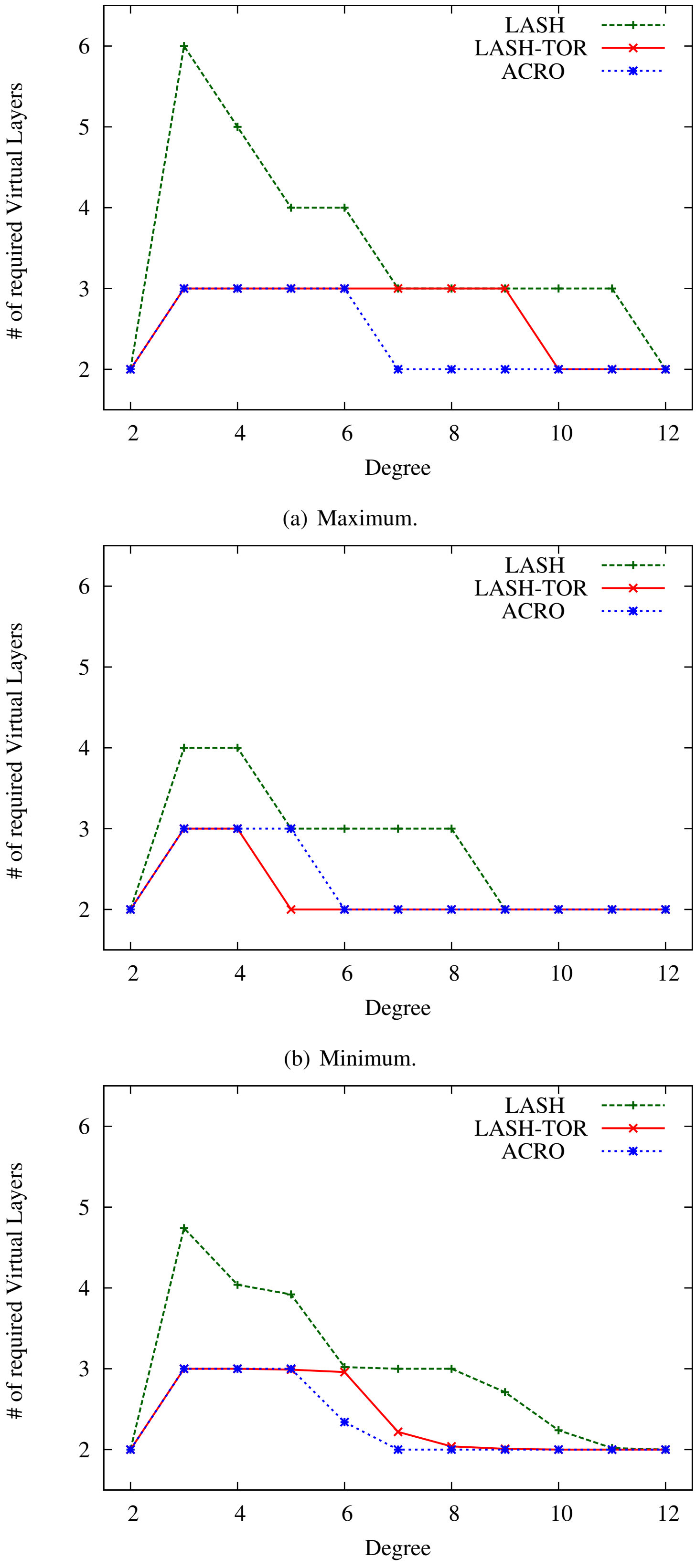
5.5.3 Algorithm Execution Time

In this section, execution time of ACRO itself is compared with that of the conventional VC assignment methods; LASH and LASH-TOR described in Section 5.5.1.1. All methods were implemented with Python scripts that use a NetworkX package. They are executed on a server with two Intel Xeon CPUs E5-2470 @ 2.30 GHz (2x8 cores) and 128GB memory. The number of nodes is varied among $|N| = 64, 256$ and 1024, and the number of degree d is set to 16. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown. Ten topologies are generated from different seeds for each node.

The average and standard deviation values are shown in Fig. 5.6. The results show that ACRO completed its VC assignment in about 30 minutes for 1,024 nodes. On the other hand, LASH and LASH-TOR consumed about 589 and 700 minutes for the same network size, respectively. In this experiment, ACRO was executed 2.3, 8.4, and 19.9 times faster than LASH-TOR for 64, 256, and 1,024 nodes, respectively. These results demonstrate the small time complexity of ACRO, previously mentioned in Section 5.5.2.

Algorithm 2 Assignment of Virtual Layers.

```
Input: $I = (N, C)$, a Routing Table
Output: a Set of Virtual Layers $L$
  for all $n \in N$ do
      Create a CDG $T_n = (C, E_n)$
  end for
  for all $n \in N$ do
      Calculate sets of weight values;
      $\{(h_{n,c}, w_{n,c}) | c \in C\}$ (See Sec. 5.3.1)
  end for
  for all $c \in C$ do
      Create an empty hash table $\mathbb{H}_c$
      for $0 \leq \Delta < D$ do
          $\mathbb{H}_c[\Delta] \leftarrow 0$
      end for
  end for
  /* Initially calculate fitness values */
  for all $n \in N$ do
      for all $c \in C$ do
          if $c$ has any parent in $T_n$ then
              $\mathbb{H}_c[h_{n,c}] \leftarrow \mathbb{H}_c[h_{n,c}] + w_{n,c}$
          end if
      end for
  end for
  for all $c \in C$ do
      $f(c) \leftarrow \max(\{\Delta | 0 \leq \Delta < D, \mathbb{H}_c[\Delta] > 0\} \cup \{0\})$
  end for
  Set boolean values $\{m(n, c) = \text{False} | n \in N, c \in C\}$
  /* Set Virtual Layers repeatedly */
  $i \leftarrow 0$
  repeat
      Create a new empty strictly ordered set $L_i$
      Create a set $U$, a copy set of $C$
      while $U \neq \emptyset$ do
          From $U$ remove $u_{\min}$ which minimizes $f(u)$,
          breaking ties by $\mathbb{H}_u[f(u)]$
          Add $u_{\min}$ to the head of $L_i$
          for all $n \in N$ do
              if $u_{\min}$ has no parent in $T_n$ then
                  $m(n, u_{\min}) \leftarrow \text{True}$
                  for all $c' \in C_{\text{child}}(n, u_{\min})$ do
                      Remove an edge $(c', u_{\min})$ in $T_n$
                      $\mathbb{H}_{c'}[h_{n,c'}] \leftarrow \mathbb{H}_{c'}[h_{n,c'}] - w_{n,c'}$
                      while $\mathbb{H}_{c'}[f(c')] = 0 \wedge f(c') > 0$ do
                          $f(c') \leftarrow f(c') - 1$
                      end while
                  end for
              end if
          end for
      end while
      Add $L_i$ to the head of $L$
      $i \leftarrow i + 1$
  until $\forall n \in N\ \forall c \in C, m(n, c) = \text{True}$
```

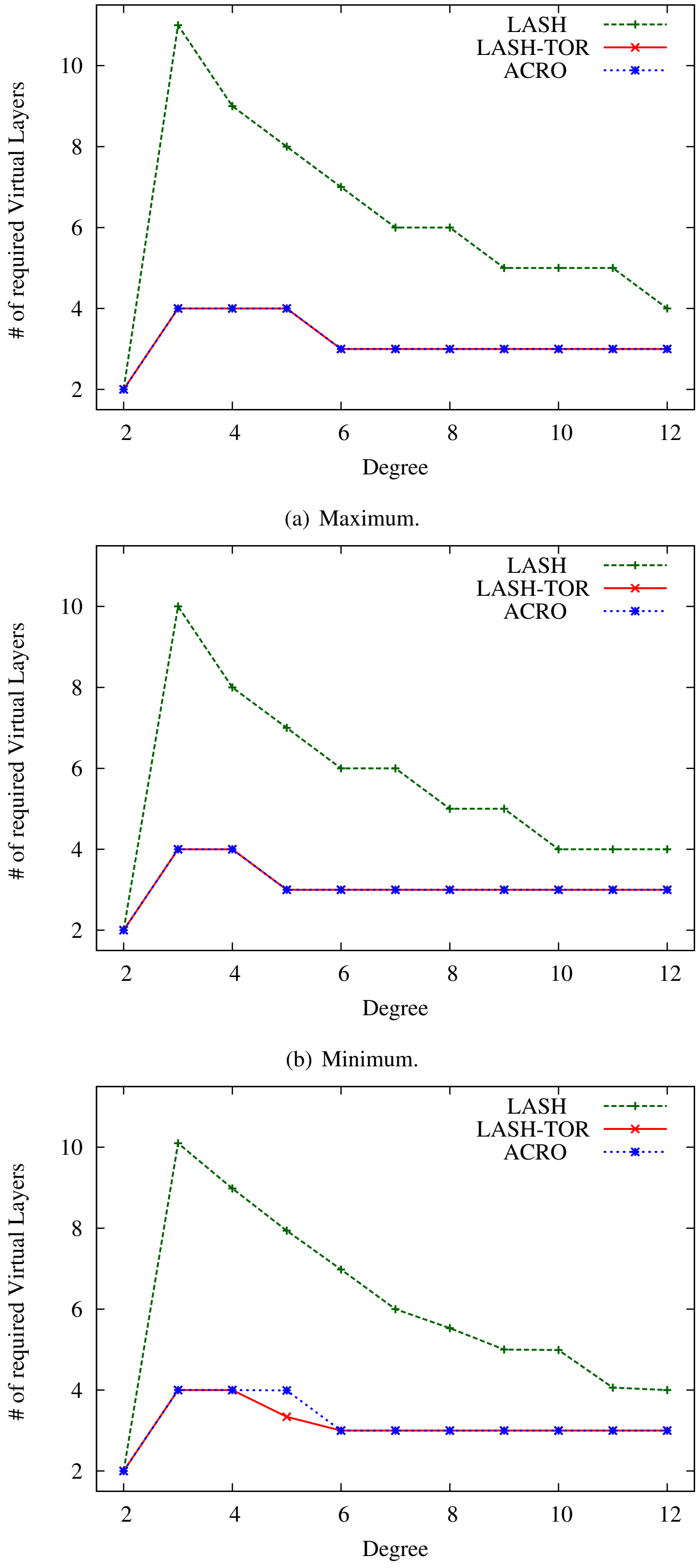


(a) Maximum.

(b) Minimum.

(c) Average.

Figure 5.4: Number of required VLs (64 nodes)



(a) Maximum.

(b) Minimum.

(c) Average.

Figure 5.5: Number of required VLs (256 nodes)

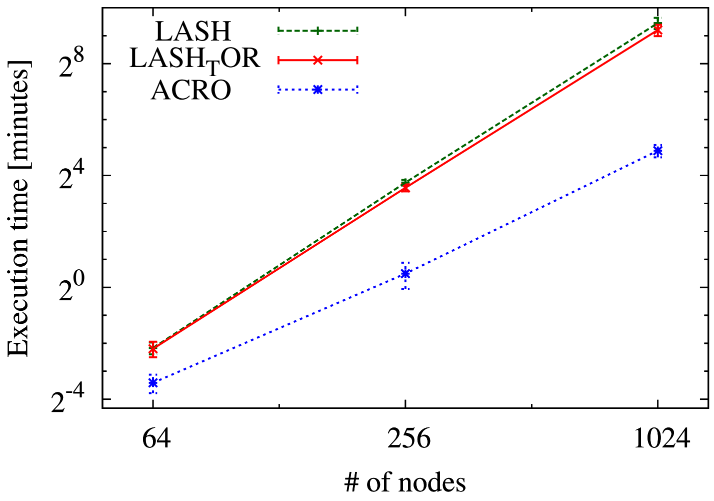


Figure 5.6: Execution time of each algorithm (d = 16).

Chapter 6

Network Simulation Results

In this chapter, the network performance of LOREN introduced in Chap. 4 with ACRO proposed in Chap. 5 is evaluated. This evaluation measures the end-to-end latency and throughput on layout-conscious random topologies.

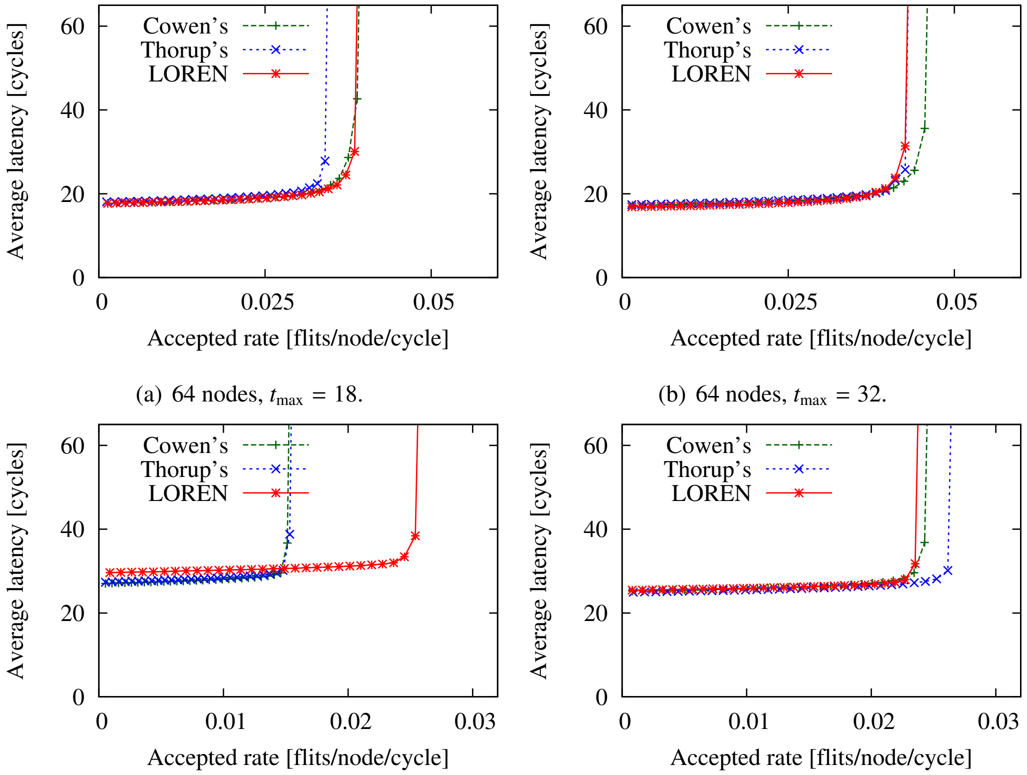
A cycle-accurate network simulator Booksim [36] is used for evaluation. For the two compact routing algorithms and LOREN algorithm, deadlock-freedom is supported with the proposed method ACRO that assigns traffics to multiple virtual channels. Network parameters for the simulation are shown in Tab. 6.1.

Figure 6.1 to 6.4 show simulation results under uniform, transpose, shuffle, and reverse traffics [37]. Due to the increased simulation time, the evaluation results for the case of more than 1,024 nodes cannot be shown. For 64 nodes, LOREN can reduce the saturation throughput by 13.2 % compared to Thorup's compact routing method in a uniform traffic with the number of table entries of $t_{\max} = 18$, as shown in FIg. 6.1(a). In this case, LOREN achieves almost the same throughput as Cowen's algorithm. However, in the same traffic with $t_{\max} = 32$, LOREN degrades the throughput by 6.6 % compared to Cowen's algorithm. In this case, Cowen's compact routing method can utilize a larger number of landmarks, which leads to distributed the traffic load and thus to improvement of the network capacity. Additionally, LOREN degrades the throughput under transpose and shuffle traffics. For these deterministic traffic patterns, LOREN fails to distribute the concentration of flows in some of links in the network. On the other hand, in a reverse traffic, LOREN can increase the throughput by 25.0 % and 14.8 % for $t_{\max} = 18$ and $t_{\max} = 32$, respectively. Moreover, LOREN achieves the lower latency than the compact routing methods in most cases with 64 nodes. It reduces the latency by up to 5.0 %.

For 1,024 nodes, LOREN can achieve the better throughput especially with the small number of table entries. In the case of a uniform traffic and $t_{\max} = 116$, the throughput is increased by 67.9 % compared to that with Cowen's algorithm, which is shown in Fig. 6.1(c). The increase is also shown in most of the other traffic patterns. As a result, LOREN develops capacity to balance traffic load. Although LOREN increases the zero-load latencies with $t_{\max} = 116$, it can fill the gap with a large number table entries of $t_{\max} = 502$.

Table 6.1: Network parameters.

Simulation period	100,000 cycles
Packet size	1 flit
Number of VCs	2 (for 64 nodes) 6 (for 1,024 nodes)
Buffer size per VC	8 flits
Number of pipeline stages	4



(a) 64 nodes, $t_{\max} = 18$.

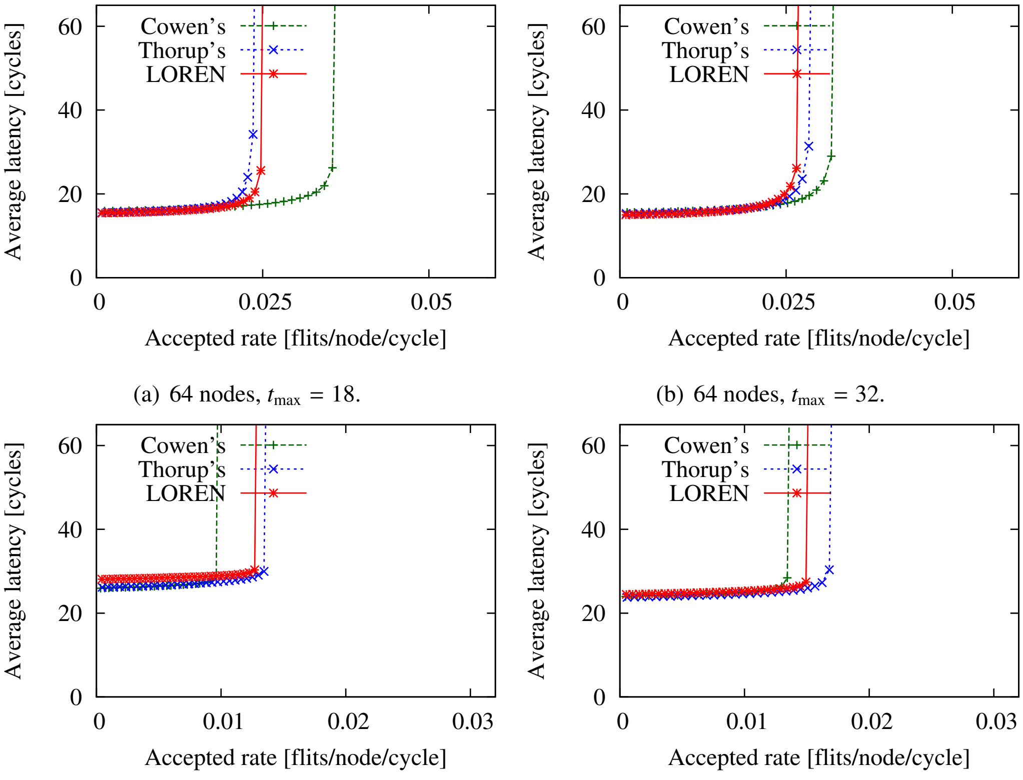
(b) 64 nodes, $t_{\max} = 32$.

(c) 1,024 nodes, $t_{\max} = 116$.

(d) 1,024 nodes, $t_{\max} = 502$.

Figure 6.1: Network performance under uniform traffic.

In summary, LOREN, which utilizes local routing information, can achieve comparable latencies to those with Cowen's algorithm, which exploits global routing information. Moreover, LOREN can avoid load concentration with the distributed routing manner, which Cowen's routing method has to face because of the large amount of flows to "landmark" nodes.



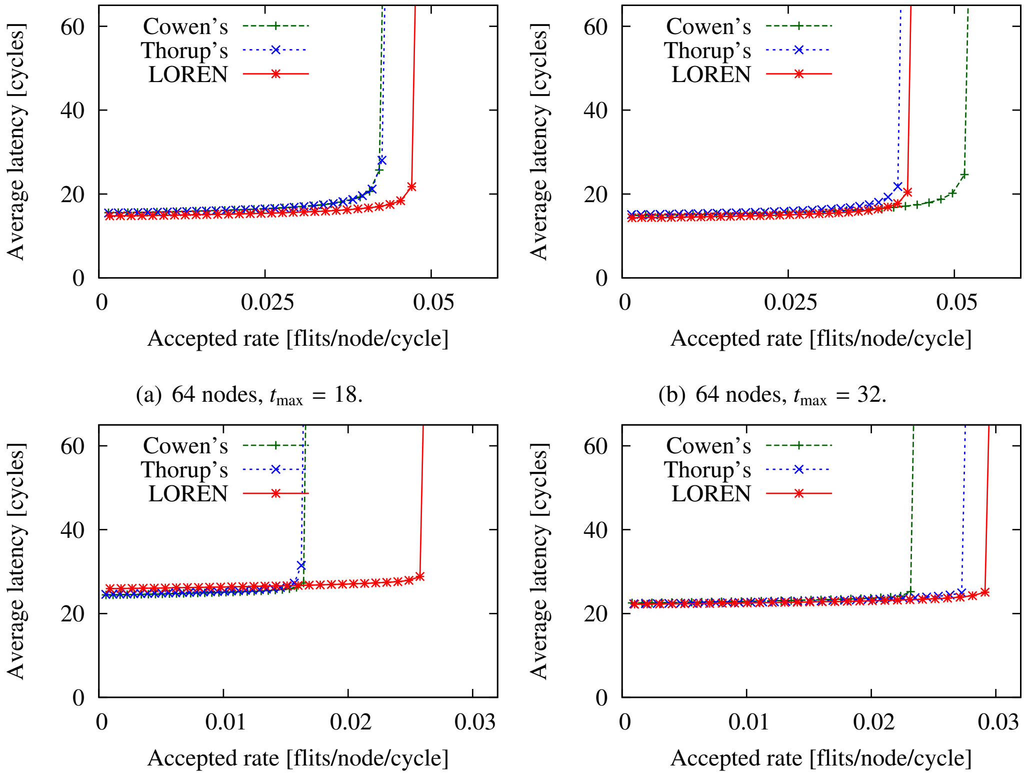
(a) 64 nodes, $t_{\max} = 18$.

(b) 64 nodes, $t_{\max} = 32$.

(c) 1,024 nodes, $t_{\max} = 116$.

(d) 1,024 nodes, $t_{\max} = 502$.

Figure 6.2: Network performance under transpose traffic.



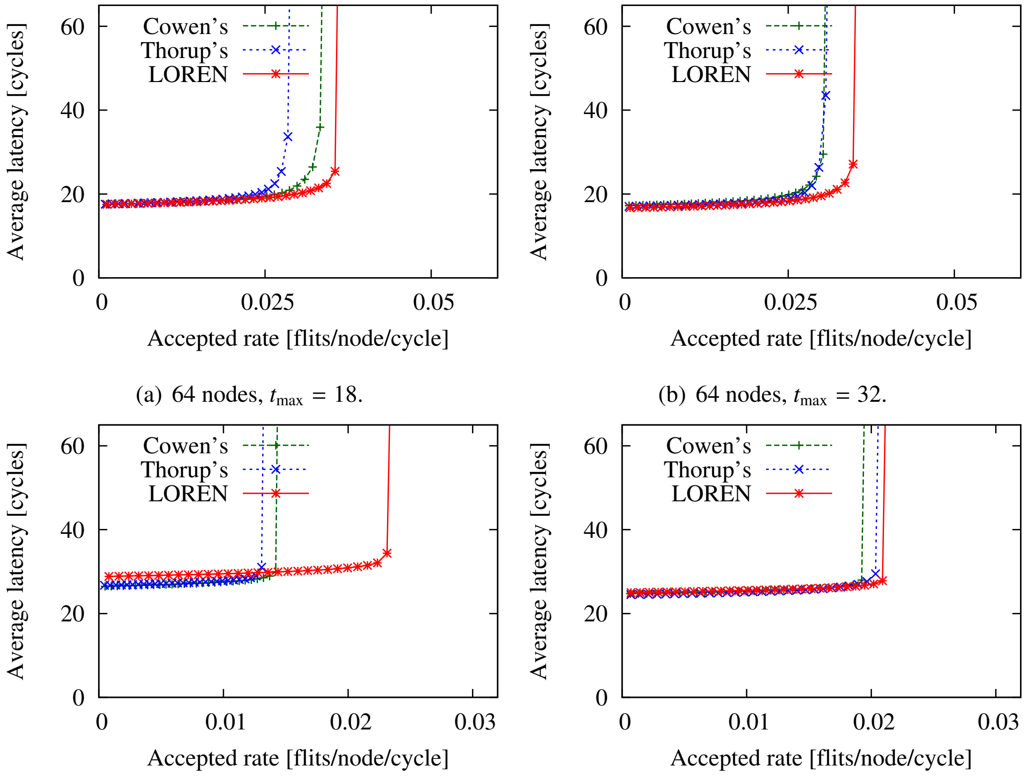
(a) 64 nodes, $t_{\max} = 18$.

(b) 64 nodes, $t_{\max} = 32$.

(c) 1,024 nodes, $t_{\max} = 116$.

(d) 1,024 nodes, $t_{\max} = 502$.

Figure 6.3: Network performance under shuffle traffic.



(a) 64 nodes, $t_{\max}$ = 18.

(b) 64 nodes, $t_{\max}$ = 32.

(c) 1,024 nodes, $t_{\max}$ = 116.

(d) 1,024 nodes, $t_{\max}$ = 502.

Figure 6.4: Network performance under reverse traffic.

Chapter 7

Conclusions

This thesis aims to achieve practical low-latency interconnection networks by introducing a scalable routing methodology for layout-conscious random topologies. Lately proposed random topologies can drastically reduce the number of hops compared to conventional regular topologies such as Torus or Fat-tree. However, they cannot be naively utilized because of a huge amount of a total cable length and the difficulty in implementation of routing tables. To provide a feasible solution, attention is focused on the layout-conscious random topologies that have random shortcut links with their lengths limited. They can achieve a comparable average number of hops between nodes while can drastically reduce the total cable length.

The firstly proposed method LOREN (Layout-Oriented Routing with Entries for Neighbors) exploits locality of connections between nodes in the layout-conscious topologies to achieve both the small number of hops between nodes and small routing table sizes required. As shown in Fig. 7.1, the proposed routing method LOREN can reduce the required table size compared to the conventional compact routing methods. Moreover, the achieved table size is significantly less than that required for shortest path routing and the upper bound for implementation with Infiniband [25]. Note that in this figure, due to the increased simulation time, the evaluation results for the case of more than 4,096 nodes by Cowen's and Thorup's compact routing methods cannot be shown.

The secondly proposed method ACRO (Assignment of Channels in Reverse Order) can be applied to arbitrary livelock-free routing methods to ensure deadlock-freedom for the path between any pair of source and destination nodes. Figure 7.2 shows the achieved trade-offs between the time complexity and the required number of VCs by ACRO and the conventional deadlock-free methodologies introduced in Sec. 3.2. As LASH-TOR [18], ACRO can reduce the number of Virtual Channels (VC) by assigning each sub-path that is derived from each path to one of the ordered VCs. Moreover, ACRO can reduce the time complexity of the algorithm by introducing a cyclic dependency check for each VC, in the same way as the recent implementation of LASH (LASH ver.2) [29].

Experimental results show that combination of LOREN and ACRO can improve the scalability and flexibility for implementation yet can reduce the network latency. The proposed VC assignment method ACRO can reduce the average number of VCs by up to 63% when compared with a conven-

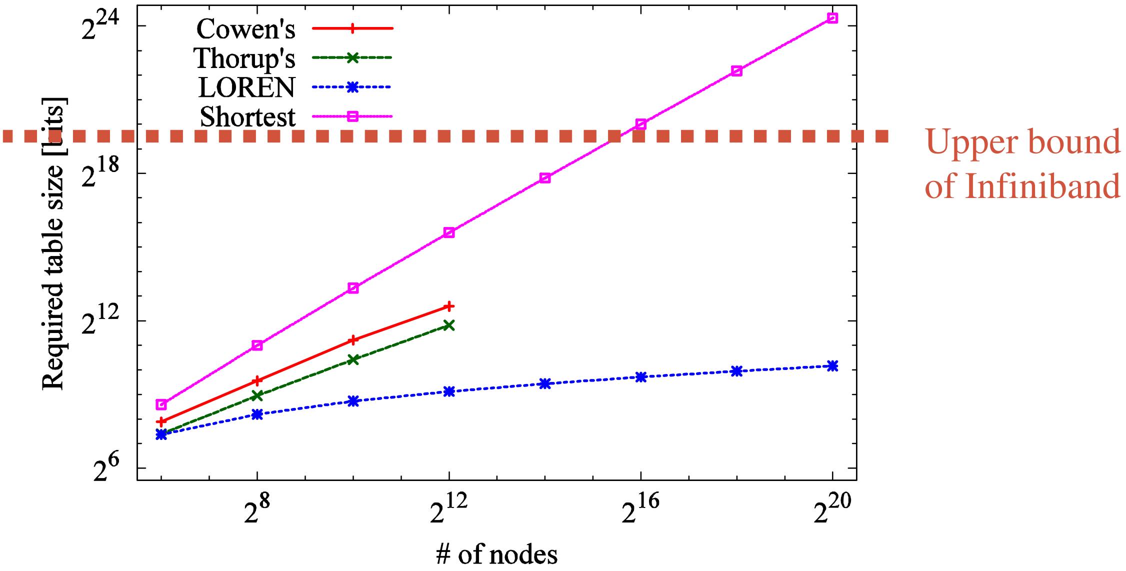


Figure 7.1: Routing table size required for the proposed algorithm LOREN and conventional algorithms in randomized networks and the upper bound for Infiniband.

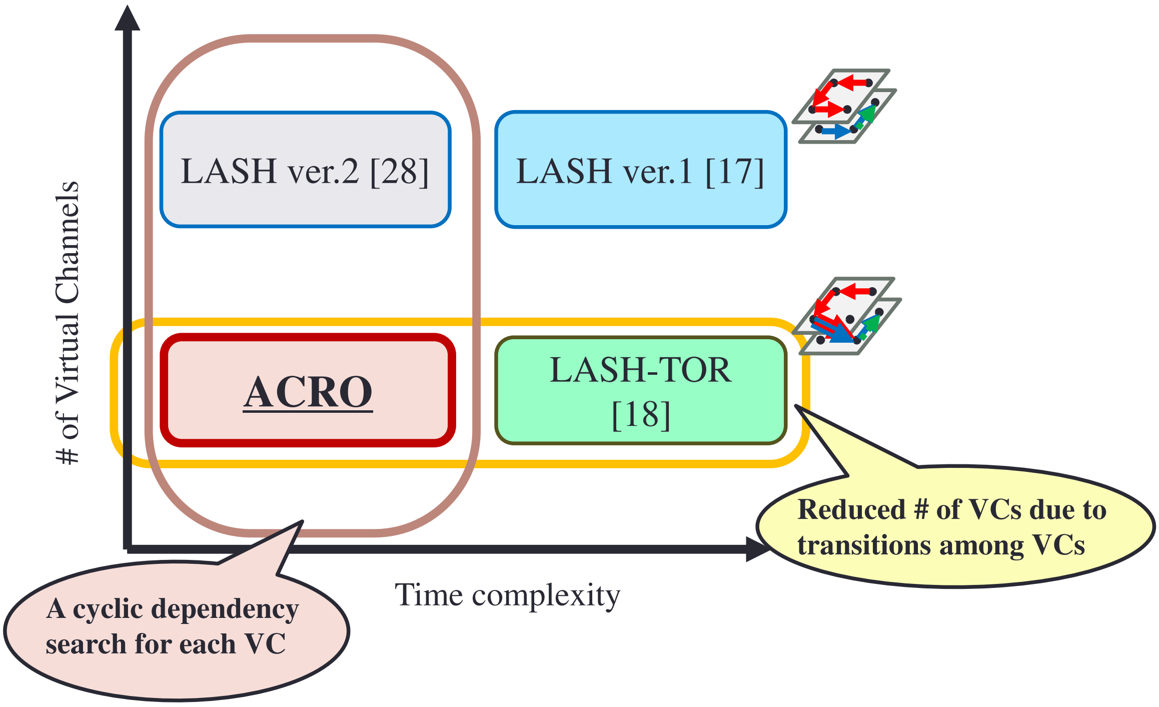

# of Virtual Channels
LASH ver.2 [28]
LASH ver.1 [17]
ACRO
LASH-TOR [18]
Reduced # of VCs due to transitions among VCs
Time complexity
A cyclic dependency search for each VC

Figure 7.2: Qualitative comparison of the proposed method ACRO and the conventional deadlock-free methodologies.

tional algorithm that has the same time complexity, while it also reduces the time complexity by a factor of $O(|N| \cdot \log |N|)$, when compared with another conventional algorithm that requires almost the same number of VCs. Moreover, the proposed routing methodology can reduce the average latencies

Table 7.1: Qualitative comparison of alternative solutions ('+': good, 'o': fair, '-': poor).

Topology	Torus	Fat-tree	Dragonfly	SlimFly	Random	LCR	LCR
Routing	DOR	Up*/Down*	MIN	MIN	MIN	MIN	LOREN+ACRO (proposed)
Latency	-	-	o	+	+	o	o
Table	+	+	o	o	-	-	o
# of VCs	+	+	o	o	-	-	o
Total Cable length	+	+	-	-	-	+	+

by 7.9 % and improve the network throughput by up to 67.9 % compared to a conventional compact routing method. Moreover, the number of required routing table entries is reduced by up to 91 %.

Table 7.1 shows a qualitative comparison of the conventional and proposed solutions. DOR (Dimension Order Routing) for Torus, up*/down* routing for Fat-tree, minimal routing (MIN) for Dragonfly and Slim Fly can achieve minimal routing and be implemented with a reduced number of table entries and VCs. However, the combinations of these routing methods and the regular topologies suffer from either the increased network latency or total cable length. Although fully-random and layout-conscious random (LCR) topologies can achieve the low latency, a conventional minimal routing method increases the number of table entries and VCs. In contrast, the union of the proposed methods LOREN and ACRO can reduce them yet can maintain the low latency.

Bibliography

[1] June 2017 | TOP500 Supercomputer Sites. `https://www.top500.org/lists/2017/06/`.

[2] K. Scott Hemmert et al. Report on Institute for Advanced Architectures and Algorithms, Interconnection Networks Workshop 2008. `http://ft.ornl.gov/doku/_media/iaaicw/iaa-ic-2008-workshop-report-v09.pdf`.

[3] J. Tomkins. Interconnects: A Buyers Point of View. ACS Workshop, Jun 2007.

[4] John Kim, William J. Dally, and Dennis Abts. Flattened Butterfly: a Cost-Efficient Topology for High-Radix Networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 126–137, Jun 2007.

[5] Wentao Bao, Binzhang Fu, Mingyu Chen, and Lixin Zhang. A High-Performance and Cost-Efficient Interconnection Network for High-Density Servers. In *Proc. of the IEEE 10th International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC)*, pages 1246–1253, Nov 2013.

[6] Maciej Besta and Torsten Hoefler. Slim Fly: A Cost Effective Low-Diameter Network Topology. In *Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 348–359, Nov 2014.

[7] Ji-Yong Shin, Bernard Wong, and Emin Gün Sirer. Small-World Datacenters. In *Proc. of the Symposium on Cloud Computing (SoCC)*, pages 2:1–2:13, Oct 2011.

[8] Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, D. Frank Hsu, and Henri Casanova. A Case for Random Shortcut Topologies for HPC Interconnects. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 177–188, Jun 2012.

[9] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking Data Centers Randomly. In *Proc. of USENIX Symposium on Network Design and Implementation (NSDI)*, pages 225–238, Apr 2012.

[10] Graph golf: The order/degree problem competition. `http://research.nii.ac.jp/graphgolf/`.

[11] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Jun 1998.

[12] Ümit Y. Ogras and Radu Marculescu. "It's a Small World After All": NoC Performance Optimization Via Long-Range Link Insertion. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 14(7):693–706, Jul 2006.

[13] Haofan Yang, Jyoti Tripathi, Natalie Enright Jerger, and Dan Gibson. Dodec: Random-Link, Low-Radix On-Chip Networks. In *Proc. of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 496–508, Dec 2014.

[14] Michihiro Koibuchi, Ikki Fujiwara, Hiroki Matsutani, and Henri Casanova. Layout-conscious Random Topologies for HPC Off-chip Interconnects. In *Proc. of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 484–495, Jun 2013.

[15] Ikki Fujiwara, Michihiro Koibuchi, Hiroki Matsutani, and Henri Casanova. Swap-And-Randomize: A Method for Building Low-Latency HPC Interconnects. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 26(7):2051–2060, Jul 2015.

[16] Jose Flich, Tor Skeie, Andres Mejia, Olav Lysne, Pedro Lopez, Antonio Robles, Jose Duato, Michihiro Koibuchi, Tomas Rokicki, and Jose Carlos Sancho. A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 23(3):405–425, Mar 2012.

[17] Tor Skeie, Olav Lysne, and Ingebjorg Theiss. Layered Shortest Path (LASH) Routing in Irregular System Area Networks. In *Proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[18] Tor Skeie, Olav Lysne, Jose Flich, Pedro Lopez, Antonio Robles, and Jose Duato. LASH-TOR: A Generic Transition-Oriented Routing Algorithm. In *Proc. of the 10th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 595–604, Jul 2004.

[19] John Kim, William J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 77–88, Jun 2008.

[20] McKay Miller and Jozef Širáň. Moore Graphs and Beyond: A survey of the Degree/Diameter Problem. *Electronic Journal of Combinatorics*, 20(2):1–92, Nov 2013.

[21] Paul R. Hafner. The Hoffman-Singleton Graph and its Automorphisms. *Journal of Algebraic Combinatorics*, (18):7–12, 2003.

[22] Paul R. Hafner. Geometric realisation of the graphs of McKay–Miller–Širáň. *Journal of Combinatorial Theory, Series B*, 90(2):223–232, Mar 2004.

[23] The degree diameter problem for general graphs. `http://combinatoricswiki.org/wiki/The_Degree_Diameter_Problem_for_General_Graphs/`.

[24] Koji Nakano, Daisuke Takafuji, Satoshi Fujita, Hiroki Matsutani, Ikki Fujiwara, and Michihiro Koibuchi. Randomly Optimized Grid Graph for Low-Latency Interconnection Networks. In *Proc. of the International Conference on Parallel Processing (ICPP)*, pages 340–349, Aug 2016.

[25] Tom Shanley. *Infiniband Network Architecture*. MINDSHARE, INC, 2003.

[26] Arthur Brady and Lenore Cowen. Compact Routing on Power Law Graphs with Additive Stretch. In *Proc. of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 119–128, Jan 2006.

[27] Lenore J. Cowen. Compact Routing with Minimum Stretch. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 255–260, Jan 1999.

[28] Mikkel Thorup and Uri Zwick. Compact Routing Schemes. In *Proc. of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, Jul 2001.

[29] Jens Domke, Torsten Hoefler, and Wolfgang E. Nagel. Deadlock-Free Oblivious Routing for Arbitrary Topologies. In *Proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 616–627, 2011.

[30] William J. Dally and Charles L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers (TC)*, C–36(5):547–553, May 1987.

[31] Dah Ming Chiu, Miriam Kadansky, Radia Perlman, John Reynders, Guy Steele, and Murat Yuksel. Deadlock-free Routing Based on Ordered Links. In *Proc. of the 27th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 62–71, 2002.

[32] Ikki Fujiwara, Michihiro Koibuchi, Hiroki Matsutani, and Henri Casanova. Skywalk: A Topology for HPC Networks with Low-Delay Switches. In *Proc. of the IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 263–272, May 2014.

[33] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks. In *Proc. of the IEEE International Conference on Cluster Computing (CLUSTER)*, pages 116–125, Sep 2008.

[34] Xin Yuan, Santosh Mahapatra, Michael Lang, and Scott Pakin. LFTI: A New Performance Metric for Assessing Interconnect Designs for Extreme-Scale HPC Systems. In *Proc. of the IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 273–282, May 2014.

[35] Vito Latora and Massimo Marchiori. Efficient Behavior of Small-World Networks. *Physical Review Letters*, 87:198701:1–198701:4, Oct 2001.

[36] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, John Kim, and William J. Dally. A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In *Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 86–96, Apr 2013.

[37] William James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

Publications

Related Papers

Journal Papers

[1] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, A Layout-Oriented Routing Method for Low-Latency HPC Networks, IEICE Transactions on Information and Systems, Vol.E100-D, No.12, pp.xx–xx, Dec 2017. (accepted for publication)

[2] Ryuta Kawano, Hiroshi Nakahara, Seiichi Tade, Ikki Fujiwara, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, A Novel Channel Assignment Method to Ensure Deadlock-Freedom for Deterministic Routing, IEICE Transactions on Information and Systems, Vol.E100-D, No.8, pp.1798–1806, Aug 2017.

International Conference Papers

[3] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, LOREN: A Scalable Routing Method for Layout-conscious Random Topologies, *Proc. of the 4th International Symposium on Computing and Networking (CANDAR'16)*, pp.9–18, Nov 2016. **Best Paper Award**

[4] Ryuta Kawano, Hiroshi Nakahara, Seiichi Tade, Ikki Fujiwara, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, ACRO: Assignment of Channels in Reverse Order to Make Arbitrary Routing Deadlock-free, *Proc. of the 15th IEEE/ACIS International Conference on Computer and Information Science (ICIS'16)*, pp.565–570, Jun 2016.

Domestic Conference Papers and Technical Reports

[5] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, A Scalable Routing Method for Random Topologies with the Link Length Limited, *Proc. of the 15th Forum on Information Technology*, Vol.1, pp.337–338, Sep 2016. (In Japanese)

[6] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, An Effective Virtual Channel Allocation Method for Deterministic Deadlock-free Routing, *IEICE Technical Reports CPSY2015-148*, Vol.115, No.518, pp.163–168, Mar 2016. (In Japanese)

[7] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, Easily implementable routing algorithms for irregular topologies in offchip interconnects, *Proc. of the 78th National Convention of IPSJ*, pp.1:15–1:16, Mar 2016. (In Japanese)

[8] Ryuta Kawano, Hiroshi Nakahara, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, A Low Latency Distributed Routing Method for Random Topologies in HPC Networks, *IEICE Technical Reports CPSY2015-103*, Vol.115, No.374, pp.105–110, Dec 2015. **IEICE ICD Young Presentation Award** (In Japanese)

Other Papers

Journal Papers

[9] Yusuke Matsushita, Hayate Okuhara, Koichiro Masuyama, Yu Fujita, Ryuta Kawano, Hideharu Amano, Body Bias Domain Partitioning Size Exploration for a Coarse Grained Reconfigurable Accelerator, IEICE Transactions on Information and Systems, Vol.E100-D, No.12, pp.xx–xx, Dec 2017. (accepted for publication)

[10] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, The Study of Low-latency On-chip Topology using Multiple Core Links, IEICE Transactions on Information and Systems, Vol.J97-D, No.3, pp.601–613, Mar 2014. (In Japanese)

International Conference Papers

[11] Ryuta Kawano, Ryota Yasudo, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, "HiRy: An Advanced Theory on Design of Deadlock-free Adaptive Routing for Arbitrary Topologies", *Proc. of the IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS'17)*, pp.xx–xx, Dec 2017. (accepted for publication)

[12] Yusuke Matsushita, Hayate Okuhara, Koichiro Masuyama, Yu Fujita, Ryuta Kawano, Hideharu Amano, Body bias grain size exploration for a coarse grained reconfigurable accelerator, *Proc. of the 26th International Conference on Field-Programmable Logic and Applications (FPL'16)*, Poster session, pp.330–333, Aug 2016.

[13] Ryuta Kawano, Seiichi Tade, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, Optimized Core-links for Low-latency NoCs, *Proc. of the 23rd Euromicro Interna-*

tional Conference on Parallel, Distributed, and Network-Based Processing (PDP'15), pp.172–176, Mar 2015.

[14] Seiichi Tade, Takahiro Kagami, Ryuta Kawano, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, A Configurable Switch Mechanism for Random NoCs, *The Poster Session at the 17th IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips XVII)*, Poster session, Poster No.15, Apr 2014. **Featured Poster Award**

[15] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, Low Latency Network Topology Using Multiple Links at Each Host, *The Poster Session at the 16th IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips XVI)*, Poster session, Poster No.18, Apr 2013.

Domestic Conference Papers and Technical Reports

[16] Hiroshi Nakahara, Daichi Fujiki, Seiichi Tade, Ryota Yasudo, Ryuta Kawano, Hiroki Matsutani, Michihiro Koibuchi, Koji Nakano, Hideharu Amano, Topology Optimization of 3D-Stacked Chips under Maxiumum Wire Length Constraint, *IEICE Technical Reports CPSY2015-104*, Vol.115, No.374, pp.111–116, Dec 2015. (In Japanese)

[17] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, Interconnect Design for Low Latency, High Topological Embeddability and Partitioning Capability by Supplementary Optical Circuit Switches, *IEICE Technical Reports CPSY2014-20 (SWoPP'14)*, Vol.114, No.155, pp.61–66, Jul 2014. (In Japanese)

[18] Seiichi Tade, Ryuta Kawano, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, Alterable uniform and random NoC through rewiring, *IEICE Technical Reports CPSY2014-22 (SWoPP'14)*, Vol.114, No.155, pp.73–78, Jul 2014. (In Japanese)

[19] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, HPC interconnect for high topological embeddability by supplementary optical circuit switches, *IEICE Technical Reports CPSY2013-111*, Vol.113, No.497, pp.253–258, Mar 2014. (In Japanese)

[20] Seiichi Tade, Takahiro Kagami, Ryuta Kawano, Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, A Configurable Switch Mechanism for Random NoCs, *IEICE Technical Reports RECONF2013-77*, Vol.113, No.418, pp.125–130, Jan 2014. (In Japanese)

[21] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, A Low Latency Topology for NoC Using Multiple Host Links, *IEICE Technical Reports CPSY2013-9*, Vol.113, No.21, pp.49–54, Apr 2013. **IEICE CPSY Young Presentation Award** (In Japanese)

[22] Ryuta Kawano, Ikki Fujiwara, Hiroki Matsutani, Hideharu Amano, Michihiro Koibuchi, Low Latency Network Topology Using Multiple Links at Each Host, *IEICE Technical Reports CPSY2012-75*, Vol.112, No.376, pp.123–128, Jan 2013. (In Japanese)