LOREN: A Scalable Routing Method for Layout-conscious Random Topologies

Ryuta Kawano[1], Hiroshi Nakahara[1], Ikki Fujiwara[2],
Hiroki Matsutani[1], Michihiro Koibuchi[2], and Hideharu Amano[1]

[1]Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan
blackbus@am.ics.keio.ac.jp

[2]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{ikki, koibuchi}@nii.ac.jp

***Abstract*—End-to-end network latency has become an important issue for parallel application on large-scale high performance computing (HPC) systems. It has been reported that randomly-connected inter-switch networks can lower the end-to-end network latency. The trade-off is a large amount of routing information. For irregular networks, minimal routing is achieved by using routing tables for all destinations in the network. In this work, a novel distributed routing method called LOREN (Layout-Oriented Routing with Entries for Neighbors) to achieve low-latency with a small routing table is proposed for irregular networks whose link length is limited. The routing tables contain both physically and topologically nearby neighbor nodes to ensure livelock-freedom and a small number of hops between nodes. Experimental results show that LOREN reduces the average latencies by 2.8% and improves the network throughput by up to 39% compared with a conventional compact routing method. Moreover, the required routing table size is reduced by up to 67%, which improves scalability and flexibility for implementation.**

I. Introduction

For large parallel application executed on the next generation high performance computing (HPC) systems, MPI communication latency should be lower than one microsecond [1], [2]. Thus, low-latency inter-switch networks are essential for such systems. Since delay in switches (e.g., about 100 nanoseconds in InfiniBand QDR) is typically larger than that in the cable and for flit injection even including serial and parallel converters, network structures with small number of hops in switch are required.

To build inter-switch networks, connections among switches via inter-switch links are modeled as topologies consisting of nodes corresponding to switches and edges representing links. Structured topologies such as torus or fat-tree are conventionally used as networks for HPC systems and Datacenters, taking into account their locality of connections and scalability. Compared with these orderly connected networks, recently proposed random shortcut topologies can drastically reduce the number of hops [3], [4], [5].

There are several challenges to support scalability for implementation of the randomized HPC networks. Unlike conventional non-random topologies, random shortcut topologies have a drawback of an increased aggregate cable length on a machine room floor. The workaround is proposed which generates random topologies with the cable length limited [6], [7]. It is reported that these layout-conscious topologies achieve a comparable average number of hops to completely irregular topologies yet reduce the total link length. Another challenge for scalability is reducing huge memory space consumed by routing table entries. For the randomly connected topologies, it is necessary to use topology-agnostic routing algorithms [8]. 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 $\mathcal{O}(N \cdot \log |N|)$ for each switch. Due to this large memory space in routing tables, the conventional algorithms for irregular networks degrade scalability for larger system sizes.

To solve this problem, attention is focused on locality of connections in the layout-conscious topologies mentioned above. 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 with completely randomized topologies. Moreover, the restriction of link lengths can reduce the number of hops between nodes that are close to each other on the floor that the topologies are placed on. 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.

The rest of our paper is organized as follows. Section II shows related work. In Section III, the problem for establishing routing methodologies is defined. In Section IV, detail of the algorithm for generation of routing tables is presented. In Section V, a routing algorithm with routing tables generated in Section IV is described. In Section VI, the proposed algorithm LOREN is evaluated and compared with the conventional compact routing method. Section VII shows some future work and conclusion.

II. Related Work

A. Low-latency Random Topologies for HPC systems

High-radix topologies which use a large scale switches with a lot of links are advantageous to reduce the latency. Flattened Butterfly topology [9] and Dragonfly topology [10] have been utilized for middle scale systems. However, for large systems with more than ten thousand nodes, such topologies require too much cost. Increasing the number of dimensions of k-ary n-cube is another way to reduce the network latency. BlueGene-Q and K-supercomputer use six or five dimensional torus. However, this approach will obviously face the limitation because of the increasing number of distant links.

Low latency direct interconnection network topologies have been theoretically researched since 1980's. Although De Bruijn graphs [11] and Star graphs [12] have lower average hop count and diameter than k-ary n-cubes, they have never been used in practical systems because of their strange looking topologies.

To achieve networks with the lowest number of hops, researchers in the field of graph theory have had interests in the concept of Moore Bound [13]. It determines the maximum number of nodes in a graph with a given degree and diameter. For high-performance networks, diameter-2 optimal or quasi-optimal graphs have been adopted, in which the number of nodes is equal or close to Moore Bound. Hoffman-Singleton graph [14], which is one of the well-known optimal graphs, is exploited in intra-server networks [15]. Moreover, McKay-Miller-Širáň (MMS) Graph [16] whose number of nodes is close to Moore Bound is utilized in Slim Fly topologies [17]. Although these graphs can drastically reduce the number of hops, they cannot be directly utilized with arbitrary numbers of nodes and degrees. Optimal topologies with given numbers of nodes and degrees have been explored, which reveals difficulty in the optimization that is in most cases based on heuristic approach [18].

As a workaround plan against this difficulty, randomized topologies for HPC systems are proposed. It is shown that random networks can achieve low latencies that are comparable with those in the optimized networks and the networks based on the concept of Moore Bound. These topologies exhibit small-world phenomenon [19]. It reduces a number of hops between nodes in proportion to $\log_d |N|$, where d and $|N|$ represent a degree of each node and a number of nodes, respectively. Moreover, for arbitrary network sizes, randomized networks can be generated with the smaller computational costs than the optimized networks. These topologies applied to inter-switch networks can improve bandwidth, scalability, and fault-tolerance of the networks. It is reported that they can be utilized to both HPC networks [3], [4], [5] and on-chip networks [20].

B. Layout-conscious Random Topologies

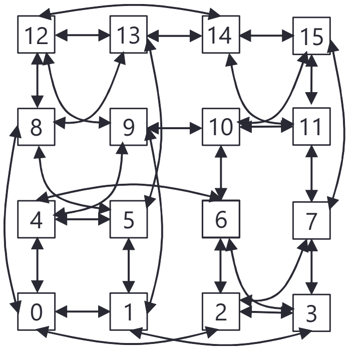


Fig. 1: Example of layout-conscious random topology $\mathrm{LRT}(|N| = 16, d = 4, r = 2)$.

Although random topologies achieve small number of hops between nodes, they need a larger amount of 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 [6], [7]. These networks achieve drastic reduction of the total cable length with minimal increase in the number of hops between nodes compared with uniform random networks.

In this work, a distributed routing method is explored for random topologies with the link length limited. The notation of the layout-conscious topology is defined as $\mathrm{LRT}(|N|, d, r)$, where $|N|$, d, and r denote the number of nodes, the degree of each node, and the maximum length of edges, respectively. Here, topologies among nodes are assumed as regular graphs, which means that all nodes in the topologies have the same degree. Fig. 1 shows an example of a layout-conscious topology $\mathrm{LRT}(16, 4, 2)$ that is placed on a $n \times n$ 2-dimensional surface. 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. All edges in this topology are connecting between two nodes so that the Manhattan distance between them, which is defined in Section III-A, is equal or less than the limitation value r.

C. Reduction of Routing Table Size

The performance of packet transfers highly depends on the implementation of routers in which packet routing information is stored and referred. For random network, a routing table indicating the output link for all destination nodes is required 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.

For layout-conscious random topologies, since the range of nodes which can be connected with direct links is limited, we can much reduce the number of entries. 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.

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 minimum increase in numbers of hops. For evaluation of this methodology, Stretch Factor is defined as the maximum rate of increase in numbers of hops. There is thus a trade-off between the size of routing table and Stretch Factor. Several methods to make compact routing tables are proposed, one of which adds a label to each node for compressing information for shortest paths. Such labeling enables a compact routing scheme called Interval Routing [21].

Another routing method called Cowen's method uses topology-dependent names for nodes, and achieves Stretch Factor less than 3 with a local routing table of size $\mathcal{O}(|N|^{2/3} \log^{4/3} |N|)$ [22]. In the method, some nodes are selected as "landmarks" that cover all nodes in a network. With Cowen's method, packets are forwarded as the following procedure.

When a current node that a packet exists 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 according to 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 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 according to the information.

In summary, with Cowen's method, a minimal path is selected in the case of nearby destination nodes, otherwise a path to one of 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.

In this work, a well load-balanced and thus high-throughput routing method called LOREN (Layout-Oriented Routing with Entries for Neighbors) is proposed for layout-conscious random topologies. It is evaluated from the viewpoints of hop counts and network throughput comparing with Cowen's compact routing method.

III. Problem Definition

A. 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 $n \times n$ 2-dimensional coordinate, and the nodes are arranged on the lattice positions. Namely, a set 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 lead to longer cable lengths and degrade the overall performance [23].

TABLE I: 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)	x- and y-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 number 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

Additionally, the minimal number of hops between nodes i and j is denoted with $h(i, j)$.

B. 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. I. 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.

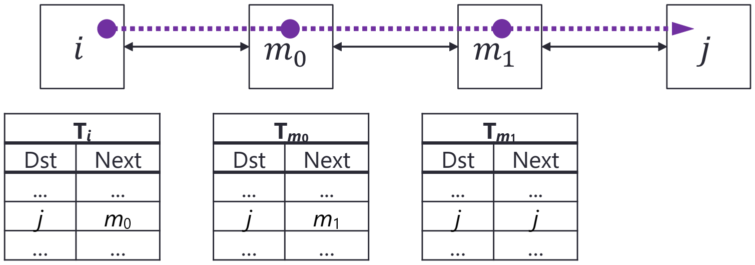
IV. Algorithm for generating Routing Table Entries in LOREN

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 Section IV-A.

In this method, reachability of packets is supported by entries for topologically and physically neighboring destination nodes as shown in Section IV-A1 and IV-A2. 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 Section IV-A3. 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 between nodes for the layout-conscious random topologies.

A. Construction of Table Entries for Shortest Paths

1) Shortest Paths between Adjacent Nodes: Routing information is provided between two nodes that are 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.



T_i	
Dst	Next
...	...
j	m_0
...	...

T_{m_0}	
Dst	Next
...	...
j	m_1
...	...

T_{m_1}	
Dst	Next
...	...
j	j
...	...

Fig. 2: Example of Routing table entries for a path from a node i to a node j, $P(i,j)$ ($|P(i,j)| = 4$).

Algorithm 1 Generation of routing tables

```
Input: $\mathrm{LRT}(|N|, d, r)$, maximum # 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 $0 \leq k \leq |P(i,j)| - 3$ 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 $i \in N$ do
      Create a spanning tree for dst $i$, $G_i$
  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
```

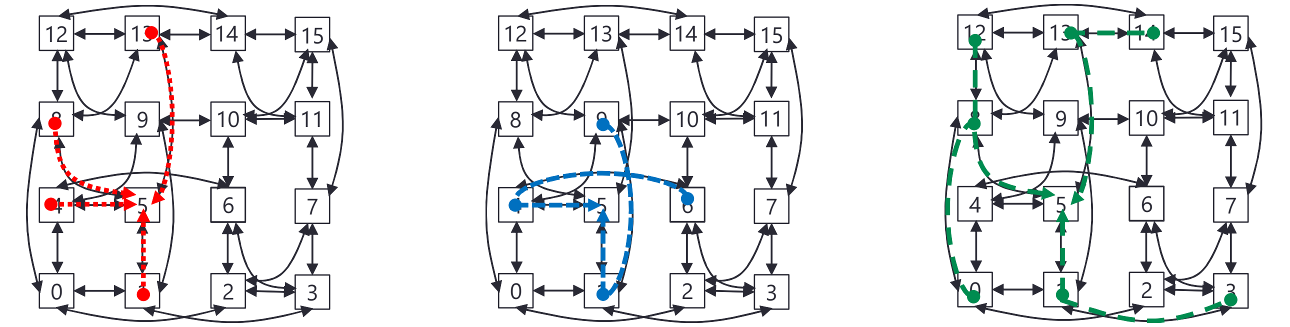
2) Shortest Paths between Neighboring Nodes on a Coordinate: On a 2-dimensional coordinate that is defined in Section III-A, a set of two nodes i and j is neighboring where the Manhattan Distance between them $\mathrm{md}(i,j)$ is equal to one. Entries for a shortest path between these two nodes are added with the following procedure.

A shortest path from a node i to a node j is represented as a consecutive nodes,

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

To all nodes in the path except a node j, entries for a destination node j and nodes of the next hop along the path are added. Namely, an entry of $<j, m_{k+1}>$ is added to each node $m_k \in \{m_0, ..., m_{|P(i,j)|-3}\}$. 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 Section IV-A1; therefore, the entry is not added again. Packets can be forwarded along this shortest path with these routing table entries in the following way. A packet whose destination is a node j injected to a node $i(= m_0)$ is forwarded to a node m_1 with reference of the entry $<j, m_1>$. Similarly, when the packet is in an intermediate node m_k for all $0 < k \leq |P(i,j)| - 3$, the entry $<j, m_{k+1}>$ is referred and the packet is routed along the shortest path towards a node j. Finally the packet reaching a node $m_{|P(i,j)-2|}$ is forwarded to a destination node j with the entry $<j, j>$. Fig. 2 shows an example of routing table entries which achieves the shortest path with four hops from a node i to a node j. Note that these entries for the shortest path from i to j are also utilized for the shortest path from the intermediate node m_k for all $0 < k \leq |P(i,j)| - 2$ towards a destination node j.

3) Shortest Paths between Distant Nodes: After generation of routing table entries for the two kinds of shortest paths mentioned in Section IV-A1 and IV-A2, there might be some empty routing entries in each node. These entries are utilized for construction of table entries 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 put addition of the entries for paths with a smaller number of hops ahead of addition of those for paths with a larger number of hops. This is achieved by the following ways. Initially spanning trees are generated for all destination nodes, which are represented as $\{G_i | i \in N\}$, each of which contains shortest paths for the destination node i. After this generation, Breadth First Search is applied simultaneously to all trees. In this search, if a visited node u in a tree G_v has an empty entry, an entry $<v, w_{\mathrm{succ}}>$ is added to the node u, 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 Section IV-A2, a packet whose destination is v in a node that contains an entry for v can be forwarded along the shortest path.



(a) Shortest paths for adjacent nodes.

(b) Shortest paths for neighboring nodes.

(c) Shortest paths for distant nodes.

Fig. 3: Example of shortest paths for a destination node #5 induced with table entries ($t_{\max} = 7$).

B. *Example of Table Entries for Destination Node*

Fig. 3 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 $t_{\max} = 7$. A starting point of a dashed arrow that is depicted with filled circle represents the node which has an entry of $<5, v_{\text{end}}>$, 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. 3a, 3b, and 3c show paths for adjacent nodes, neighboring nodes, and distant nodes as described in Section IV-A1, IV-A2, and IV-A3, 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 Section IV-A2 and IV-A3. It is notable that the new entries that would be duplicated by already added entries for the adjacent nodes are not added for these shortest paths.

V. Routing Algorithm in LOREN

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

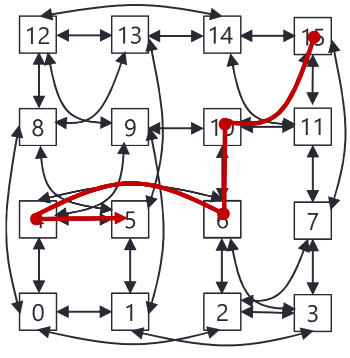
A node of the next hop for a packet in a node u whose destination node is v is determined with the following procedure. Among routing table entries in the node u, the entry $<v_{\text{dst}}, v_{\text{next}}>$ that minimizes the Manhattan Distance between nodes v_{dst} and v, $\text{md}(v_{\text{dst}},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

	T15	
	Dst	Next
	7	7
	10	10
	11	11
	14	14
	2	7
	3	7
✓	**6**	*10*

(a) Routing table in node #15.

	T10	
	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.

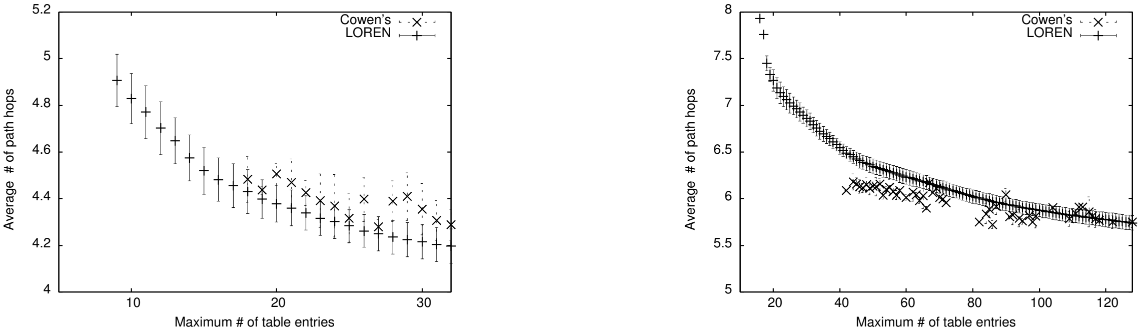
Fig. 4: Example of routing from node #15 to #5 ($t_{\max} = 7$).

An example of routing packets is shown in Fig. 4. A given topology and generated table entries are the same as the example in Fig. 3. Let a packet be routed from a node #15 to a #5. A routing table that a node #15 has is shown in Fig. 4a. 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, $\text{md}(6, 5) = 1$. The packet is thus forwarded to a node #10. A routing table in a node #10 is shown in Fig. 4b. In the same way, an entry $<6, 6>$ is selected that has the smallest Manhattan Distance. Note that an entry $<9, 9>$ also has the smallest Manhattan Distance. In this case, an entry $<6, 6>$ is selected according to 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. 3. The packet is then forwarded along this path.

The livelock-freedom in LOREN is proved as follows.

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

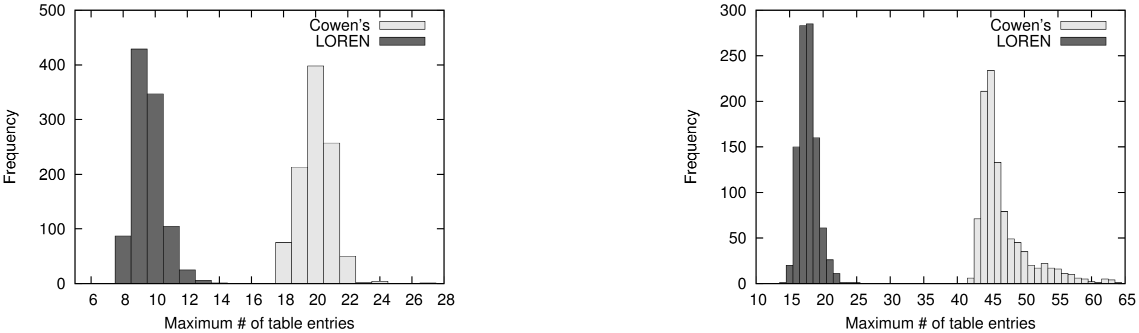
1) $\text{md}(\text{tg}_v(u), v) > \text{md}(\text{tg}_v(u'), v)$



(a) 64 nodes.

(b) 256 nodes.

Fig. 5: Maximum number of entries $t_{\max}$ and achieved average path length.



(a) 64 nodes.

(b) 256 nodes.

Fig. 6: Maximum number of table entries required for algorithms.

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 us consider a packet in u for a destination v that is forwarded to u', where $v \neq u'$ is satisfied. If $h(u, \mathrm{tg}_v(u)) > 0$, 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 Section IV-A2. In both cases, the order $u \succ_v u'$ 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 Section. IV-A1. □

VI. Evaluation

In this section, the proposed routing method LOREN is evaluated and compared with Cowen's compact routing method [22] introduced in Section II-C. Routing methods are applied to the layout-conscious random topologies, LRTs, that are defined in Section II-B. For the evaluations, $\mathrm{LRT}(64, 4, 2)$ and $\mathrm{LRT}(256, 4, 4)$ are adopted. Namely, the degree of each node is set to four. Moreover, the maximum link lengths are set to two and four for 64- and 256-node networks, respectively.

A. Table Entry Size and Average Path Length

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

Fig. 5a and Fig. 5b show the results for 64- and 256-node networks, respectively. 10 different layout-conscious random topologies are generated and used for evaluation. Evaluated values in the figures represent the average and standard deviation of those from different topologies. In these figures, legends of the conventional and proposed methods are represented as "Cowen's" and "LOREN", respectively. These legends are used in subsequent evaluations.

For both network sizes, LOREN achieves the average path lengths almost proportional to Cowen's conventional compact routing algorithm. For 256-node networks, relatively small numbers of table entries $t_{\max}$ degrade the path lengths achieved with LOREN. In this evaluation, the smallest value of $t_{\max}$ that Cowen's method can take is 42. In the same case, LOREN increases the average path length by 6.6%.

Cowen's conventional compact routing can reduce the number of hops between nodes with global information that each node contains as "landmark" nodes. On the other hand, in LOREN, shorter paths are not taken between nodes because of a small amount of local information that each node has. As the value of $t_{\max}$ increases, LOREN achieves comparable or smaller numbers of path lengths. For 64 nodes, LOREN can reduce the average path hops by up to 4.2%. Moreover, for 256 nodes, it can reduce the average number of path lengths by up to 2.8%. These result from small-world phenomena in the layout-conscious random topologies that can be utilized with larger local information in each node.

B. Minimal Number of Required Table Entries

As shown in Fig. 5, the minimum number of $t_{\max}$ that LOREN can take is smaller than that for Cowen's algorithm. Hence it is notable that LOREN improves the adaptation to various numbers of table sizes given. In this section, this flexibility is evaluated and compared with that of Cowen's algorithm in detail.

As described in Section VI, Cowen's algorithm varies 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 algorithm can take for given topologies. The LOREN algorithm constructs three kinds of shortest paths that mentioned in Section IV-A. Among them, those between distant nodes which are defined in Section IV-A3 are constructed using redundant empty entries, that is, they are optional in the algorithm. By contrast, those between adjacent nodes and between neighboring nodes, which are described in Section IV-A1 and IV-A2, 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.

Fig. 6a and Fig. 6b show distributions of the minimal number of required table entries for 1,000 64- and 256-node layout-conscious random topologies, respectively. These results show that LOREN achieves the small number of table entries with the same variation compared with Cowen's algorithm. For 64 nodes, it reduces the minimum and average number of required table entries by 56% and 52%, respectively. Moreover, it reduces the minimum and average by 67% and 62% for 256 nodes, respectively.

In this evaluation, the smallest numbers of $t_{\max}$ that Cowen's algorithm can take are 18 and 42 for 64 and 256 nodes, respectively. In the rest of evaluation, the values of $t_{\max}$ are set to both the minimum number in Cowen's algorithm and half of $|N|$. Namely, the maximum numbers of table entries are set to $t_{\max} = 18, 32$ for 64 nodes, and $t_{\max} = 42, 128$ for 256 nodes.

C. Network Simulation

A cycle-accurate network simulator Booksim [24] is used for evaluation. For both Cowen's routing algorithm and LOREN algorithm, deadlock-freedom is supported with a method to assign traffics to multiple virtual channels [25]. This assignment method is based on the assignment technique used in LASH-TOR [26]. Network parameters for simulation are shown in Tab. II.

TABLE II: Network parameters.

Simulation period	100,000 cycles
Packet size	1 flit
Number of VCs	4
Buffer size per VC	8 flits
Number of pipeline stages	4

Fig. 7 to Fig. 10 show simulation results under uniform, transpose, shuffle, and reverse traffics [27]. For 64 nodes, LOREN degrades the saturation throughput by 21% in a reverse traffic compared with Cowen's algorithm, with the table entry size of $t_{\max} = 32$, as shown in Fig. 10b. This arises from large numbers of hops in this traffic. Each node can contain only local information in LOREN, which increases the number of hops and thus the buffer occupancy with packets. Another result for 64 nodes is that Cowen's algorithm introduces instability in the saturation throughput, which depends on traffic patterns. When the value of $t_{\max}$ changes from 18 to 32 in an uniform traffic, Cowen's algorithm and LOREN can improve the saturation throughput by 40% and 38%, respectively. However, in the same case except adoption of a transpose and shuffle traffic, Cowen's algorithm degrades the throughput by 27% and 26%, respectively. On the other hand, LOREN can achieve the better throughput with larger numbers of table entries, regardless of the applied traffic. Moreover, LOREN achieves the lower latency than Cowen's algorithm in most cases with 64 nodes. It reduces the latency by up to 5.0%.

For 256 nodes, LOREN can achieve the better throughput in most cases. In the case of an uniform traffic and $t_{\max} = 42$, it increases the throughput by 39% compared with Cowen's algorithm. The increase is also shown in other traffic patterns. As a result, LOREN develops capacity to balance traffic load. It is also shown that achieved latencies with LOREN are comparable with those with Cowen's algorithm in most cases.

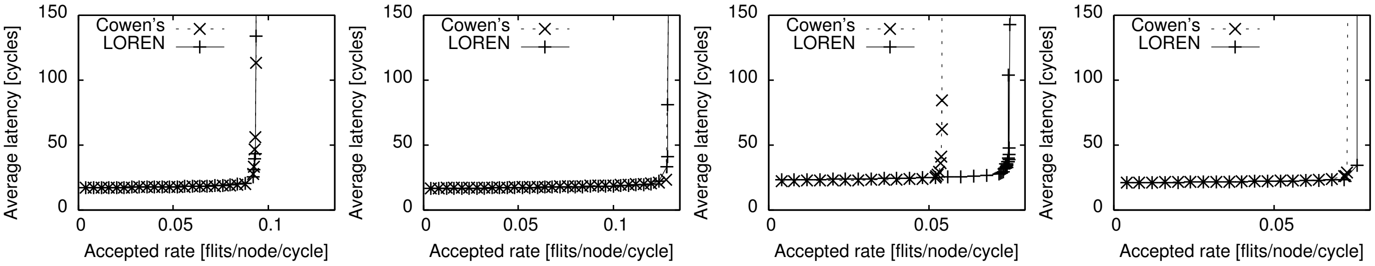
In summary, LOREN, which utilizes local routing information, can achieve comparable latencies to those with Cowen's algorithm, which exploits global routing information. Moreover, LOREN can avoid load concentration with the distributed routing manner, which Cowen's routing method has to face because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

D. Performance under Randomly Imbalanced Traffic

In Section VI-C, uniform and synthetic traffics balanced for all nodes in the network are adopted for evaluation. In this section, randomly imbalanced traffics are applied to LOREN and Cowen's method.

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.

- hotspot(β): The value $0 \leq \beta \leq 1$ denotes probability for each node to set a "hot-spot" node as a destination node.



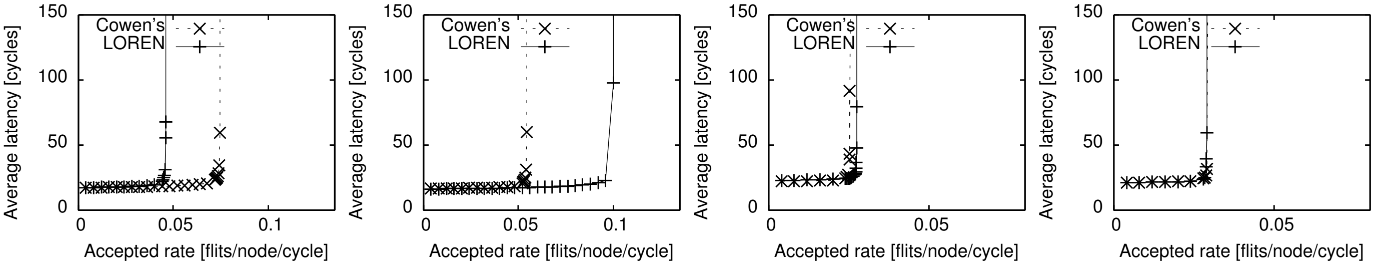
(a) 64 nodes, $t_{\max} = 18$.

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

(c) 256 nodes, $t_{\max} = 42$.

(d) 256 nodes, $t_{\max} = 128$.

Fig. 7: Network performance under uniform traffic.



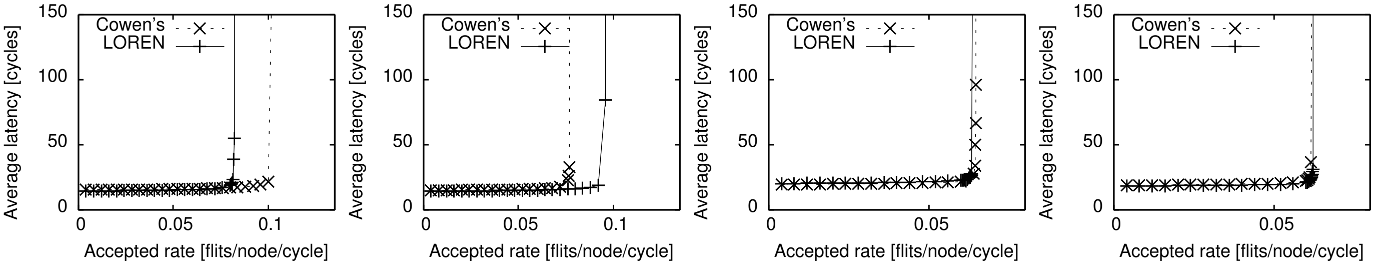
(a) 64 nodes, $t_{\max} = 18$.

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

(c) 256 nodes, $t_{\max} = 42$.

(d) 256 nodes, $t_{\max} = 128$.

Fig. 8: Network performance under transpose traffic.



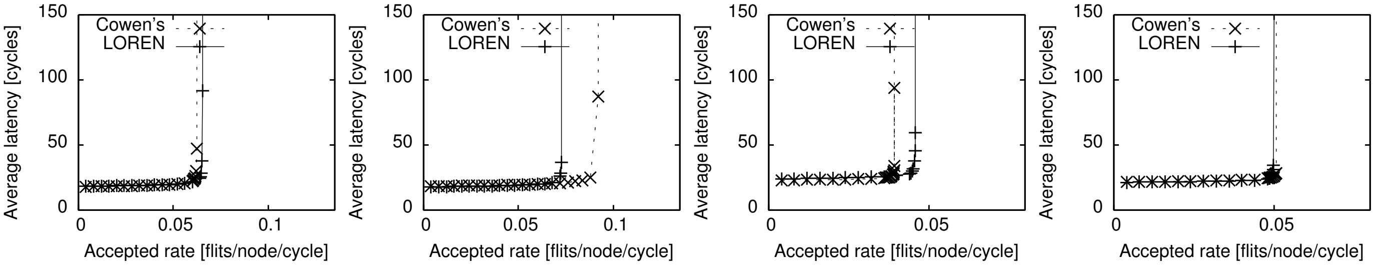
(a) 64 nodes, $t_{\max} = 18$.

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

(c) 256 nodes, $t_{\max} = 42$.

(d) 256 nodes, $t_{\max} = 128$.

Fig. 9: Network performance under shuffle traffic.



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

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

(c) 256 nodes, $t_{\max} = 42$.

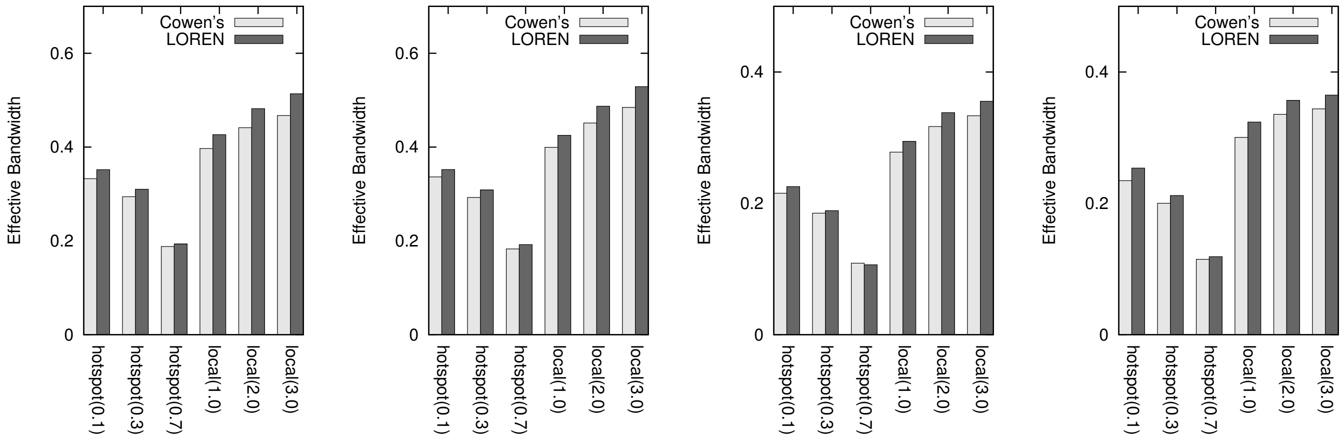
(d) 256 nodes, $t_{\max} = 128$.

Fig. 10: Network performance under reverse traffic.

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 proportional 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 $P' := \{P'(i, j)|(i, j) \in S\}$.



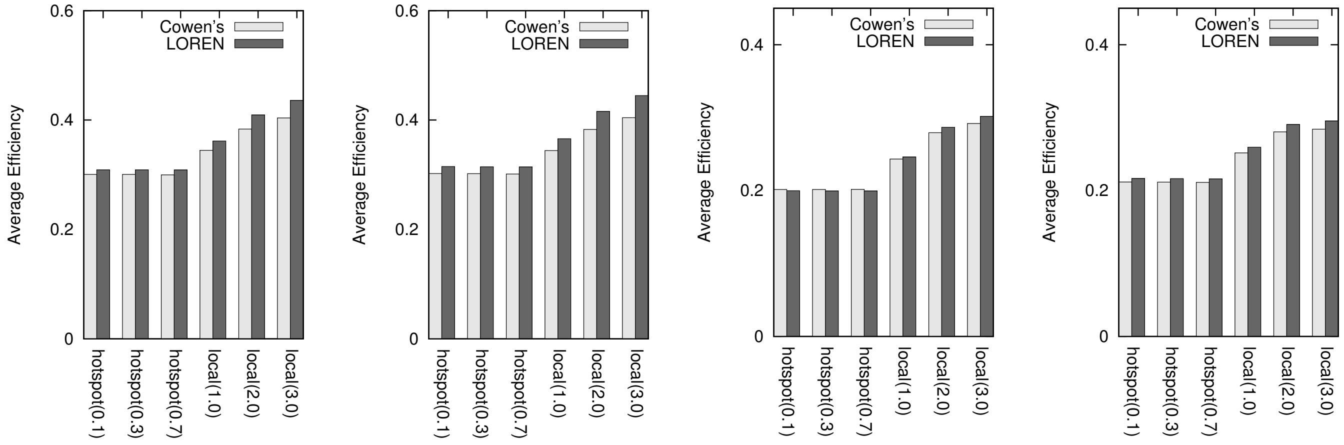
(a) 64 nodes, $t_{\max} = 18$.

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

(c) 256 nodes, $t_{\max} = 42$.

(d) 256 nodes, $t_{\max} = 128$.

Fig. 11: Effective Bandwidth under imbalanced traffic.



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

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

(c) 256 nodes, $t_{\max} = 42$.

(d) 256 nodes, $t_{\max} = 128$.

Fig. 12: Average efficiency under imbalanced traffic.

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 [28], [29].

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

In this evaluation, 1,000 random traffic patterns for each imbalanced traffic are generated, and the average effective bandwidth is calculated.

Fig. 11 shows the results. It is shown that LOREN can achieve the better performance than Cowen's algorithm except in the case of $\mathrm{hotspot}(0.7)$ and $t_{\max} = 42$ for 256 nodes. As shown in Section VI, LOREN increases the number of hops between nodes in this case, which leads to a large number of edges that each path contains and thus to high degree of congestion. On the other hand, LOREN effectively improves the bandwidth especially for the localized traffic. 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 9.1% and 6.1% for 64- and 256-node networks, respectively.

3) Average Efficiency: Efficiency for a path between nodes i and j is defined as a multiplicative inverse of the path length, $h(i,j)^{-1}$ [30]. 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 Section VI-D2, 1,000 random patterns are generated for each traffic and the average is calculated.

Fig. 12 shows that LOREN can improve the average efficiency by up to 3.3%. The improvement in the average efficiency is slightly smaller than that in the effective bandwidth as shown in Section VI-D2. 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 unbalanced traffic.

VII. Future Work and Conclusion

In this work, a scalable, low-latency, and high-bandwidth routing method, LOREN (Layout-Oriented Routing with Entries for Neighbors), for layout-conscious random topologies is proposed. This method exploits the irregularity and locality in the topologies to achieve both lower average numbers of hops between nodes and small routing table sizes. The entries in routing tables of each node are constructed so that they induce shortest paths between nodes nearby on both the topology and the coordinate. Experimental results show that LOREN can reduce the number of table entries required in each node by up to 67% for 256-node networks, compared with the conventional compact routing method. Furthermore, it can reduce the number of hops by up to 2.8%. Results obtained with network simulation show that LOREN can improve the saturation throughput by up to 39%. Moreover, it can increase the application-oriented bandwidth for the localized traffic by up to 9.1%.

In this work, generation of routing table entries and the routing algorithm with these entries are mainly focused on. For practical interconnection networks for HPC systems, it is also an important issue about implementation of LOREN on commercial switching fabrics. It can be realized with help of MPLS or OpenFlow technologies. The detailed methodology for the implementation is left for future work.

Acknowledgment A part of this work was supported by JSPS KAKENHI Grant Number 15J03374. I would like to offer my special thanks to Prof. Koji Nakano.

References

[1] 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.

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

[3] J.-Y. Shin, B. Wong, and E. G. Sirer, “Small-World Datacenters,” in *Proc. of the Symposium on Cloud Computing (SoCC)*, Oct 2011, pp. 2:1–2:13.

[4] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, “A Case for Random Shortcut Topologies for HPC Interconnects,” in *Proc. of the International Symposium on Computer Architecture (ISCA)*, Jun 2012, pp. 177–188.

[5] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, “Jellyfish: Networking Data Centers Randomly,” in *Proc. of USENIX Symposium on Network Design and Implementation (NSDI)*, Apr 2012, pp. 225–238.

[6] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, “Layout-conscious Random Topologies for HPC Off-chip Interconnects,” in *Proc. of the International Symposium on High Performance Computer Architecture (HPCA)*, Jun 2013, pp. 484–495.

[7] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova, “Swap-And-Randomize: A Method for Building Low-Latency HPC Interconnects,” *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, no. 7, pp. 2051–2060, Jul 2015.

[8] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho, “A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms,” *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 3, pp. 405–425, Mar 2012.

[9] J. Kim, W. J. Dally, and D. Abts, “Flattened Butterfly: a Cost-Efficient Topology for High-Radix Networks,” in *Proc. of the International Symposium on Computer Architecture (ISCA)*, Jun 2007, pp. 126–137.

[10] J. Kim, W. J. Dally, S. Scott, and D. Abts, “Technology-Driven, Highly-Scalable Dragonfly Topology,” in *Proc. of the International Symposium on Computer Architecture (ISCA)*, Jun 2008, pp. 77–88.

[11] M. R. Samatham and D. K. Pradhan, “The De Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI,” *IEEE Transactions on Computers (TC)*, vol. 38, no. 4, pp. 567–581, Apr 1989.

[12] S. B. Akers, B. Krishnamurthy, and D. Harel, “The Star Graph: An Attractive Alternative to the n-Cube,” in *Proc. of the International Conference on Parallel Processing (ICPP)*, Jan 1987, pp. 393–400.

[13] M. Miller and J. Širáň, “Moore Graphs and Beyond: A survey of the Degree/Diameter Problem,” *Electronic Journal of Combinatorics*, vol. 20, no. 2, pp. 1–92, Nov 2013.

[14] P. R. Hafner, “The Hoffman-Singleton Graph and its Automorphisms,” *Journal of Algebraic Combinatorics*, no. 18, pp. 7–12, 2003.

[15] W. Bao, B. Fu, M. Chen, and L. 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)*, Nov 2013, pp. 1246–1253.

[16] P. R. Hafner, “Geometric realisation of the graphs of McKay–Miller–Širáň,” *Journal of Combinatorial Theory, Series B*, vol. 90, no. 2, pp. 223–232, Mar 2004.

[17] M. Besta and T. 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)*, Nov 2014, pp. 348–359.

[18] “Graph golf: The order/degree problem competition,” http://research.nii.ac.jp/graphgolf/.

[19] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, Jun 1998.

[20] Ü. Y. Ogras and R. 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)*, vol. 14, no. 7, pp. 693–706, Jul 2006.

[21] C. Gavoille, “A survey on interval routing,” *Theoretical Computer Science*, vol. 245, no. 2, pp. 217–253, Aug 2000.

[22] L. J. Cowen, “Compact Routing with Minimum Stretch,” in *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Jan 1999, pp. 255–260.

[23] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova, “Skywalk: A Topology for HPC Networks with Low-Delay Switches,” in *Proc. of the IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*, May 2014, pp. 263–272.

[24] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W. 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)*, Apr 2013, pp. 86–96.

[25] R. Kawano, H. Nakahara, S. Tade, I. Fujiwara, H. Matsutani, M. Koibuchi, and H. Amano, “ACRO: Assignment of Channels in Reverse Order to Make Arbitrary Routing Deadlock-free,” in *Proc. of the 15th IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, Jun 2016, pp. 565–570.

[26] T. Skeie, O. Lysne, J. Flich, P. Lopez, A. Robles, and J. Duato, “LASH-TOR: A Generic Transition-Oriented Routing Algorithm,” in *Proc. of the 10th International Conference on Parallel and Distributed Systems (ICPADS)*, Jul 2004, pp. 595–604.

[27] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[28] T. Hoefler, T. Schneider, and A. 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)*, Sep 2008, pp. 116–125.

[29] X. Yuan, S. Mahapatra, M. Lang, and S. 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)*, May 2014, pp. 273–282.

[30] V. Latora and M. Marchiori, “Efficient Behavior of Small-World Networks,” *Physical Review Letters*, vol. 87, pp. 198 701:1–198 701:4, Oct 2001.