ACRO: Assignment of Channels in Reverse Order to Make Arbitrary Routing Deadlock-free

Ryuta Kawano[1], Hiroshi Nakahara[1], Seiichi Tade[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*—Distributed routing methods with small routing tables are scalable design on irregular networks for large-scale High Performance Computing (HPC) systems. Recently proposed compact routing methods, however, do not guarantee deadlock-freedom. Cyclic channel dependencies on arbitrary routing are typically removed with multiple Virtual Channels (VCs). However, challenges still remain to provide good trade-offs between a number of required VCs and a time complexity of an algorithm for assignment of VCs to paths. In this work, a novel algorithm ACRO is proposed for enriching arbitrary routing functions with deadlock-freedom with a reasonable number of VCs and a time complexity. Experimental results show that ACRO can reduce the average number of required VCs by up to 63% compared with the conventional algorithm that has the same time complexity. Furthermore, ACRO reduces a time complexity by a factor of $\mathcal{O}(|N| \cdot \log |N|)$ compared with that of the other conventional algorithm that needs almost the same number of VCs.**

I. Introduction

For large parallel applications executed on next generation High Performance Computing (HPC) systems, MPI communication latency should be lower than one microsecond [1], [2]. There is thus a strong need for low-latency inter-switch networks for these systems. 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. Therefore, to achieve low latency, inter-switch topologies should have low diameter and low average shortest path length, both of which can be measured in numbers of switch hops.

Compared with the conventional Torus or Fat-tree networks, recently proposed random shortcut topologies can drastically reduce the number of hops [3], [4], [5]. It is reported that such topologies can be efficiently applied to inter-switch networks for HPC systems and Datacenters. For these topologies, it is necessary to use topology-agnostic routing algorithms [6]. To implement such algorithms, each switch must have forwarding table entries for all of the destination nodes, which degrades scalability for larger system sizes.

A problem of dealing with trade-offs between the number of hops and the number of forwarding table entries has been well discussed by researchers in the field of distributed computing. Such problem is called Compact Routing [7]. Recent technologies of MPLS or OpenFlow enable such routing methods to be utilized for HPC networks. These algorithms only support livelock-freedom and do not offer deadlock-freedom. Therefore, enriching livelock-free routing, including compact routing methods, with deadlock-freedom would be needed to apply it to practical HPC networks.

TABLE I: Trade-offs on conventional algorithms.

Routing	# of required VLs	Time Complexity
LASH	9 for 256 SWs	$\mathcal{O}(\|N\|^2)$
LASH-TOR	4 for 256 SWs	$\mathcal{O}(\|N\|^3)$

In this work, in order to support deadlock-freedom for arbitrary routing methods, multiple Virtual Channels (VCs) for each physical channel are exploited. With a given topology and the routing table obtained from a livelock-free routing algorithm assumed as inputs, VC assignment to paths is generated by our new methodology, which has a small time complexity yet with the same number of VCs compared with the conventional method. Our new methodology is referred to as Assignment of Channels in Reverse Order (ACRO).

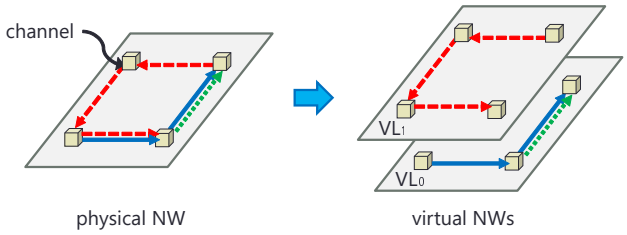
The rest of our paper is organized as follows. Section II shows related work. In Section III, the detailed ACRO algorithm is presented. In Section IV, the ACRO algorithm is evaluated and compared with the conventional VC assignment methods. In Section V, discussion about the implementation on Infiniband is presented. Section VI shows some future work and conclusion.

II. Related Work

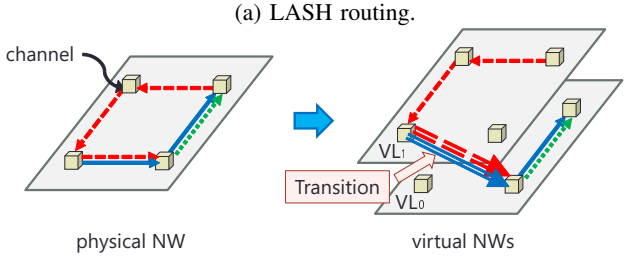
A. Virtually Layered Networks

Some deadlock-free routing methods exploit multiple Virtual Channels (VCs) to break cyclic channel dependencies. One of them is LASH routing [8], [9], in which each VC belongs to one Virtual Layer (VL). Each source-and-destination pair is assigned to one of VLs, as shown in Fig. 1a. Each VL contains subset of a set of all paths, which does not form cyclic channel dependencies. This routing method supports deadlock-free minimal paths at the cost of relatively large number of VLs. For 256 nodes, LASH uses up to 9 VLs. To avoid its weakness, an extended method, LASH-TOR [10] was proposed. It can reduce the number of required VLs by permitting transitions among ordered VLs in descending order,

978-1-5090-0806-3/16/$31.00 copyright 2016 IEEE
ICIS 2016, June 26-29, 2016, Okayama, Japan
Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.



(a) LASH routing.



(b) LASH-TOR routing.

Fig. 1: Routings with virtual layers.

as shown in Fig. 1b. Only 4 VLs for 256 nodes are required to support deadlock-free minimal routing with LASH-TOR. The trade-off is its time complexity, $\mathcal{O}(|N|^3)$, which makes it quite unrealistic for large-sized networks. We summarize the trade-offs on these routing algorithms in Table I. Although these methods mainly focus on shortest path routing, they can be extended for arbitrary routing including non-minimal routing by calculating paths obtained from the routing and treating them as inputs to the methods. In this work, multiple VLs are exploited to support deadlock-freedom in the same way.

B. Ordered Channels

To avoid cyclic channel dependencies in each VL, the ordered channels are introduced. A previous work [11] proves the following theorem.

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

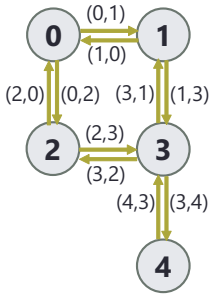
This can be proved by a topological sort for a Channel Dependency Graph (CDG) induced by a set of paths [12]. In this work, a heuristic is introduced for the channel ordering in each VL to maximize the number of paths and the length of each path routed within the VL. By combining this heuristic approach and the transitions among VLs as mentioned in Section II-A, the number of required VLs is minimized.

III. ACRO for Deadlock-free Routing

A. Problem Definition

According to the general models [11], network configuration 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.*



(a) Topology

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

Fig. 2: Example of given inputs.

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 our methodology, a topology and a routing table shown in Fig. 2 are given as an interconnection network and a routing function, respectively. The routing table in Fig. 2b 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 $c_{\mathrm{local},\mathrm{n_d}}$.

The definition of a VL L_i of a network I is the same manner as that in LASH-TOR [10], 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 the sorted physical channels of I such that every path within L_i is restricted to use the corresponding VCs in decreasing order.

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-and-destination pair, the path can reach the destination by using channels in decreasing order within each L_i and transitions among layers in decreasing order within L.

B. CDG Generation for Each Destination

In the conventional implementation of LASH-TOR [10], at least a cyclic dependency check must be done for a path. Since a cyclic dependency search has a time complexity $\mathcal{O}(|C| + |E|)$, the minimum time complexity per VL becomes approximately $\mathcal{O}(|N|^3)$. In the recent improvement on LASH [9], only a cyclic dependency check per VL is needed. This can be done by initially adding all paths to a CDG and checking 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 $\mathcal{O}(|N|^2)$. 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 detail of ACRO is shown in Alg. 1. A set of paths from a set of nodes N to a destination node n_{d} is determined as

Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.

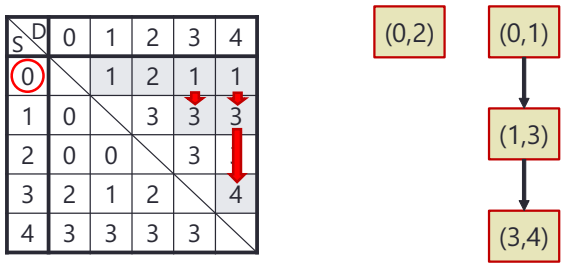
Algorithm 1 Assignment of Virtual Layers.

Input: $I = (N, C)$, a Routing Table
Output: a Set of Virtual Layers L

```
for all n ∈ N do
    Create a CDG T_n = (C, E_n)
    for all c ∈ C do
        Calculate a weight value w(n, c)
        m(n, c) ← False
    end for
end for
/* Set an initial number of Virtual Layers */
i ← 0
repeat
    Create a new empty strictly ordered set L_i
    Create a new empty set U
    for all c ∈ C do
        Add c to U
        for all n ∈ N do
            Calculate a fitness value f(n, c)
        end for
        Calculate the sum of fitness values F(c)
    end for
    while U ≠ ∅ do
        From U remove c_min which minimizes F(c)
        Add c_min to the head of L_i
        for all n ∈ N do
            if c_min has no parent in T_n then
                m(n, c_min) ← True
                for all c'' ∈ C_child(n, c_min) do
                    Remove an edge (c'', c_min) in T_n
                    F(c'') ← F(c'') − w(n, c'')
                end for
            end if
        end for
    end while
    Add L_i to the head of L
    i ← i + 1
until ∀n ∈ N ∀c ∈ C, m(n, c) = True
```

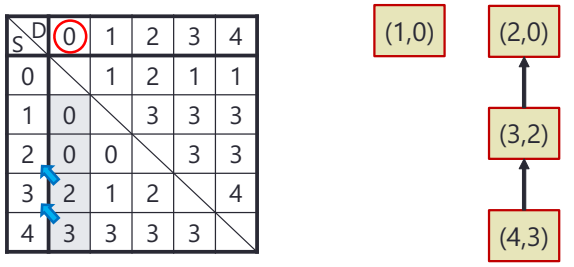
$T'_{n_\mathrm{d}} = (N, C_{n_\mathrm{d}})$, satisfying $C_{n_\mathrm{d}} \subset C$. T'_{n_d} forms a directed tree whose root is n_d, and all edges are directed toward the direction of n_d. T'_{n_d} produces a Channel Dependency Graph (CDG) for the destination n_d. Here, it is represented as $T_{n_\mathrm{d}} = (C, E_{n_\mathrm{d}})$, where E_{n_d} denotes a set of the channel dependencies. T_{n_d} is a set of directed trees as shown in Fig. 4b, and the node of the CDG is corresponding to 'channel.' For avoiding the confusion, a node of the CDG is called '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 is due to the following reasons. As shown in Fig. 3b, 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. 3a). Therefore, the time complexity becomes $\mathcal{O}(|N| \cdot \log |N|)$ for each source node. Note that $\mathcal{O}(\log |N|)$ comes from the characteristics of irregular networks [13].



(a) Routing Table (b) CDG for src. #0

Fig. 3: Creation of CDG for src. #0.



(a) Routing Table (b) CDG for dst. #0

Fig. 4: Creation of CDG for dst. #0.

On the other hand, as shown in Fig. 4b, if a CDG is generated for each destination node, only a column of the table is needed to be referred (Fig. 4a). As a result, the time complexity becomes $\mathcal{O}(|N|)$ for each destination node.

C. Heuristic Approach to Reduce VLs

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

$C_{\mathrm{child}}(n, c)$ is a set of channels which are children of the channel c in T_n. In T_n, $w(n, c)$ is defined as a weight of a channel c, which is calculated by the following recursive formula.

$$w(n,c) = \begin{cases} 1 & (\text{if } C_{\mathrm{child}}(n,c) = \emptyset) \\ \displaystyle\sum_{c' \in C_{\mathrm{child}}(n,c)} |N| \cdot w(n,c') & (\text{otherwise}) \end{cases}$$

This value increases depending mainly on the distance of the channel c from the leaf, and secondarily on the number of descendants. This calculation is done at the beginning of the algorithm.

Immediately after generating a new VL, a fitness value of c in T_n, $f(n, c)$, is calculated. This value is defined as follows.

$$f(n,c) = \begin{cases} 0 & (\text{if } c \text{ has no parent in } T_n) \\ w(n,c) & (\text{otherwise}) \end{cases}$$

Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.

The value of 0 implies that c is a root in one of the trees in T_n, and thus the smallest order is desirable to be assigned into c. Furthermore, the sum of fitness values for c, $F(c)$, is calculated as follows.

$$F(c) = \sum_{n \in N} f(n, c)$$

D. 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. 1. The termination condition of the algorithm is that all channels $c \in C$ are reachable in T_n for all $n \in N$.

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 channel $c_{\min}$ which minimizes the value of $F(c)$ is selected from unassigned channels, and added to the head of L_i. For each CDG T_n, if the channel $c_{\min}$ does not have a parent in T_n, the following procedures are done. Since $c_{\min}$ is reachable in T_n, the boolean value of 'True' is assigned to $m(n, c_{\min})$. Furthermore, edges from all the children of $c_{\min}$ to $c_{\min}$ are removed if exists.

This deletion of the edges denotes that the dependency between the child of $c_{\min}$ and $c_{\min}$ 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 $c_{\min}$. This means that the dependency is dissolved within the current VL L_i. Otherwise, the order of the child is surely smaller than that of $c_{\min}$, 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 $c_{\min}$ in L_i.

E. Proof of Deadlock-freedom

Theorem 2. *The VC and VL assignment with ACRO can enrich a given topology and a routing table with deadlock-freedom.*

Proof. Each VC that belongs to c in a VL L_i is labeled with a two-digit number $(i, \mathrm{Idx}(i, c))_{|C|}$, where $\mathrm{Idx}(i, c)$ is the order of c in L_i and $(i, j)_{|C|} = i \cdot |C| + j$. Given these labeling to VCs, the proposed algorithm in Alg. 1 establishes a route of the consecutive channels which are labeled in strictly decreasing order for every source-and-destination pair. With the notation of Theorem 1, the resultant routing is deadlock-free. □

IV. Evaluation

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 III, a topology of switches and a routing table are given as inputs. Note that the routing table takes source and current nodes, and returns a destination node.

A. Number of Required VLs

In this section, the impact of the network size and the node degree to the number of VLs is analyzed. Note that the degree is corresponding to the number of ports of a switch. In this evaluation, the VC assignment algorithms in LASH and LASH-TOR are implemented as follows.

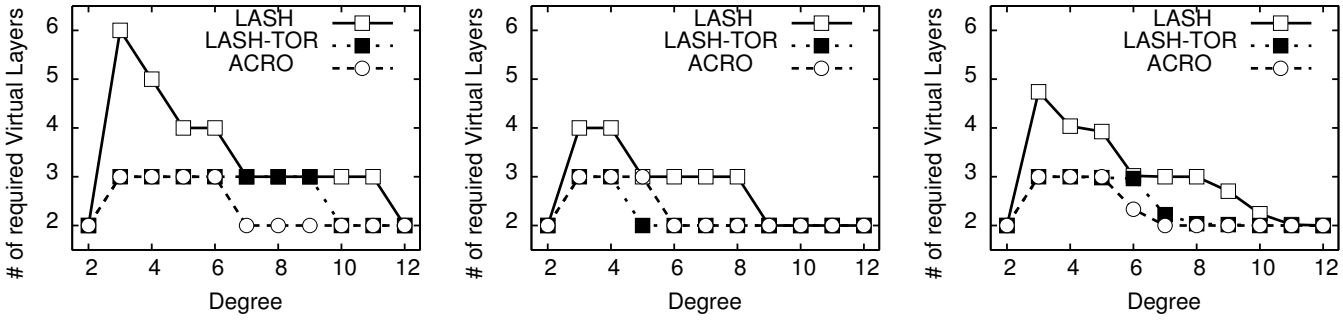
1) VC Assignment Methodology in LASH: Let $\mathrm{G}_{\mathrm{CD},i}$ be a CDG created by a set of paths in L_i. For a given path between a source node n_{s} and a destination node n_{d}, the algorithm searches a set of VLs L to find $L_i \in L$. The path induces the channel dependencies, which could 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.

2) VC Assignment Methodology in LASH-TOR: Let G_{CD} be a set of CDGs created by sets of paths in a set of VLs L. For the same inputs as Section IV-A1, 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 $\mathrm{G}_{\mathrm{CD},i} \in \mathrm{G}_{\mathrm{CD}}$ without generating a cycle of dependencies within the VL $L_i \in L'$ 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. This technique is not used in this evaluation because a routing table is given as an input. It means that the alternative paths are forbidden.

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 $|N| = 64, 256$. 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. In the original LASH and LASH-TOR routing, only a topology is assumed as an input. This enables them to select a minimal path from multiple minimal paths if exist so that the number of VLs is minimized. This path selection does not occur in this evaluation because of the given routing table.

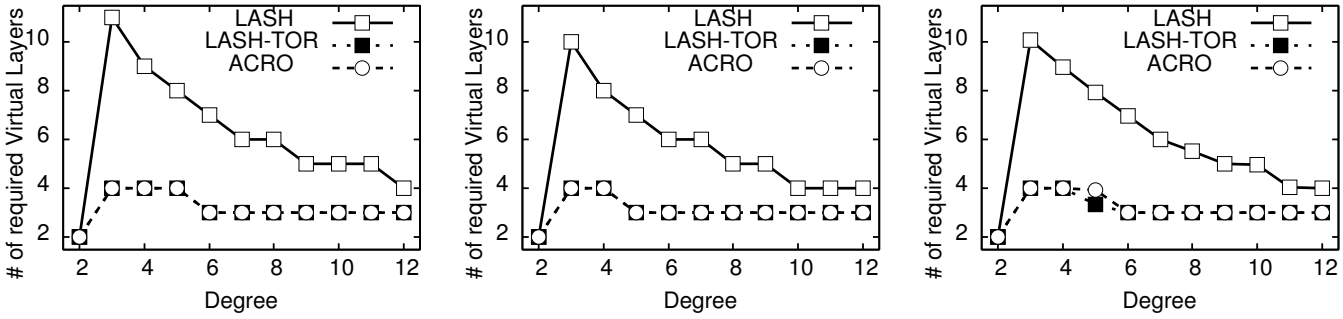
Fig. 5 and Fig. 6 show the maximum, minimum, and average numbers of required VLs for 64- and 256-node topologies. These results show that ACRO efficiently reduces the number of VLs compared with the VC assignment methodology in LASH routing. For 64-node topologies, it reduces the average and maximum 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. The additional result is that the heuristic used in ACRO achieves almost the same number of required VLs as the VC assignment methodology

Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.



(a) Maximum. (b) Minimum. (c) Average.

Fig. 5: Number of required VLs (64 nodes)



(a) Maximum. (b) Minimum. (c) Average.

Fig. 6: Number of required VLs (256 nodes)

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 our 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.

B. Time and Memory Complexity

The original algorithm for the VC assignment in LASH [8] is accelerated by the recent implementation [9]. 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*

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

while the memory complexity is

$$\mathcal{O}(\nabla \cdot h(I) \cdot |N|^2 + \nabla \cdot (|C| + |E|)).$$

In this proposition, ∇ and $h(I)$ are a number of required VLs and diameter of a topology, respectively. These parameters are used in the propositions to be hereinafter described. On the other hand, the time and memory complexities of VC assignment algorithm in LASH-TOR [10] are as follows.

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

$$\mathcal{O}(|N|^2 \cdot (h(I) \cdot \nabla \cdot (|C| + |E|)))$$

while the memory complexity is

$$\mathcal{O}(\nabla \cdot (|C| + |E|))$$

From the proposed ACRO algorithm shown in Alg. 1, the time and memory complexities are summarized as follows. The generation of CDGs for all the destinations has a time complexity of $\mathcal{O}(|N|^2)$ and a memory complexity of $\mathcal{O}(|C| \cdot |N|)$. A time complexity to calculate all the weight values $w(n, c)$ is $\mathcal{O}(|N| \cdot (|C| + |E|))$, while a memory complexity is $\mathcal{O}(|C| \cdot |N|)$. The summation of fitness values to evaluate $F(c)$ has a time complexity of $\mathcal{O}(|C| \cdot |N|)$ and a memory complexity of $\mathcal{O}(|C|)$. Moreover, the selection of the channel which minimizes the fitness function $F(c)$ and the modification of $F(c)$ need time complexities of $\mathcal{O}(|C|)$ and $\mathcal{O}(|N|^2)$ in total, respectively.

These findings as mentioned from the above is followed by these propositions.

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

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

while its memory complexity is

$$\mathcal{O}(|C| \cdot |N| + \nabla \cdot |C|).$$

Given that a topology is randomly generated and the degree d is quite smaller than the network size $|N|$, the proportionalities $h(I) \propto \log|N|$, $|C| \propto |N|$, and $|E| \propto |N|$ are satisfied [13]. From these proportionalities, it can be said that ACRO reduces a time complexity by a factor of

Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.

$\mathcal{O}(|N| \cdot \log |N|)$ compared with that of the VC assignment algorithm in LASH-TOR yet with almost the same number of required VLs.

V. Implementation on Infiniband

In this section, implementation of ACRO on Infiniband is discussed and compared with that of the conventional VC assignment methodologies in LASH and LASH-TOR.

On Infiniband networks, a packet only includes the information of a destination node as a destination LID (Local IDentifier) in the header. Moreover, the forwarding table in each switch refers only the LID of the packets to determine the output channel. In LASH and LASH-TOR algorithms, to reduce the numbers of required VCs, one of the minimal paths for a given source-and-destination pair is selected. Therefore, packets which have the same destination node may use the different output ports in a switch. For this reason, LASH and LASH-TOR cannot be implemented with the naive manner because of the routing specification of Infiniband switches mentioned before.

A workaround plan [14] against this problem exploits LMC (LID Mask Control) to use multiple virtual switches for a single physical switch. This implementation lacks scalability because one sub-net can use approximately 48,000 LIDs at maximum. On the other hand, ACRO takes a routing table as an input. It means that packets for the same destination node must use the same output port regardless of the source node. Thus, the LIDs can be assigned to physical switches in a one-to-one correspondence manner. Hence it can be said that ACRO is more suitable for implementation on Infiniband than the conventional LASH and LASH-TOR.

Another merit of ACRO is that a VL used by a packet is determined only by a destination node of the packet and an output channel. However, the SLtoVL mapping table in Infiniband switches does not take the destination node of the packet as an input. It instead takes the SL (Service Level) of the packet. One possible solution to implement VL assignment with ACRO is to embed the information of the destination node to the SL. Although the number of SLs is limited to 16, the different destination nodes can share the same SL unless there is any conflict in the SL-to-VL mapping. This sharing could reduce the number of required SLs. The detailed methodology is our future work.

VI. Future Work and Conclusion

The remaining work is as follows. Dealing with the fault tolerance is one of big challenges from the viewpoint of practical HPC networks. It would be solved by extension of our methodology for multi-path routing. Here, we mainly focused on the number of VLs and the time complexity of the algorithm. The methodology for load balancing among VCs also remains as another challenge.

We conclude this paper as the following summary. An algorithm to make arbitrary routing methods for irregular networks deadlock-free was proposed with a reasonable number of virtual channels and a time complexity. In the algorithm, VCs are assigned to paths from the destination node to the source node. In order to remove cyclic dependencies with a number of VLs as small as possible, a heuristic is introduced. The proposed VC assignment algorithm, ACRO, has a quite small time complexity yet requires almost the same number of VLs compared with that in LASH-TOR.

Experimental results show that ACRO supports both the reduced number of VLs, equivalent to that in LASH-TOR, and the time complexity as small as LASH at the same time. We believe that ACRO is the versatile and scalable scheme, in terms of both implementation and time complexity, to enrich arbitrary routing with deadlock-freedom for the future HPC networks.

Acknowledgment A part of this work was supported by JSPS KAKENHI Grant Number JP 15J03374.

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, 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)*, 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)*, 2012, p. 17.

[6] 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," in *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 3, Mar. 2012, pp. 405–425.

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

[8] T. Skeie, O. Lysne, and I. Theiss, "Layered Shortest Path (LASH) Routing in Irregular System Area Networks," in *Proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[9] J. Domke, T. Hoefler, and W. E. Nagel, "Deadlock-Free Oblivious Routing for Arbitrary Topologies," in *Proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2011, pp. 616–627.

[10] 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)*, 2004, pp. 595–604.

[11] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers (TC)*, 1987.

[12] D. M. Chiu, M. Kadansky, R. Perlman, J. Reynders, G. Steele, and M. Yuksel, "Deadlock-free Routing Based on Ordered Links," in *Proc. of the 27th Annual IEEE Conference on Local Computer Networks (LCN)*, 2002, pp. 62–71.

[13] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[14] O. Lysne, "Deadlock Free Network Routing," May 2010, US Patent 7,724,674.

Authorized licensed use limited to: IEEE Xplore. Downloaded on June 01,2024 at 21:19:45 UTC from IEEE Xplore. Restrictions apply.