k-Optimized Path Routing for High-Throughput Data Center Networks

Ryuta Kawano, Ryota Yasudo, Hiroki Matsutani, and Hideharu Amano
Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan
blackbus@am.ics.keio.ac.jp

***Abstract*—Network throughput has become an important issue for big-data analysis on Warehouse-Scale Computing (WSC) systems. It has been reported that randomly-connected inter-switch networks can enlarge the network throughput. For irregular networks, a multi-path routing method called k-shortest path routing is conventionally utilized. However, it cannot efficiently exploit longer-than-shortest paths that would be detour paths to avoid bottlenecks. In this work, a novel routing method called k-optimized path routing to achieve high throughput is proposed for irregular networks. We introduce a heuristic to select detour paths that can avoid bottlenecks in the network to improve the network throughput. Experimental results show that the proposed k-optimized path routing can improve the throughput by up to 133 % compared to the conventional k-shortest path routing. Moreover, it can improve the network bandwidth while maintaining the low network latency.**

***Index Terms*—Interconnection Networks, Warehouse-Scale Computing, Data Centers.**

I. Introduction

Recently, the size of data centers for big-data processing has grown rapidly. They form a new computer class called *warehouse-scale* computers which will provide more than hundreds of thousands of nodes. Conventional interconnection networks for such a huge scale system is difficult to be formed with economic commodity switches. Also, large throughput to cope with a large request level parallelism is difficult to be satisfied.

Novel-class random networks have been proposed for such warehouse-scale computers. A ring network with random short-cut links drastically reduces the latency between nodes by the small-world effect [1]. Jellyfish [2] using a random regular graph can achieve larger throughput than that of Fat-Trees [3], which are commonly used in data centers. Since the throughput is more important than latency for homogeneous warehouse-scale computers, interconnection networks using random regular graphs are useful.

Such random regular graphs ensure its large throughput by multiple paths between nodes. However, the throughput is also dependent on applied packet routing algorithms. Conventional *equal cost multiple paths* routing (*ECMP*) [4] cannot make use of the variety of paths in the random graph, and often degrades the throughput. A routing algorithm called *k-shortest path routing* [5] has been proposed to utilize high throughput of random regular graphs. This method uses the shortest k paths between two nodes, and achieves comparative or better throughput than Fat-Trees using ECMP. However, as the number of the paths grows, the table size is also increased. Thus, there is some difficulty on the implementation for the large size graph.

This paper explores methods for choosing multiple paths with their number fixed to maximize the throughput. A linear-programming model has been proposed for obtaining the selection probability of paths which can maximize the worst-case throughput [6]. On the other hand, we obtain the maximum worst-case throughput when m paths (m is sufficiently larger than k) are used between two nodes with the linear-programming. Here, since the selection probability of each path can be calculated, k paths with higher probability can be selected. We propose *k-optimized path routing*, which maximizes the worst-case throughput by using these k paths.

The rest of the paper is organized as follows. Section II overviews conventional data center networks and high-throughput routing methods. Section III shows a conventional framework to find the optimum throughput. Section IV describes our novel high-throughput routing called k-optimized path routing for irregular networks. In Section V, our proposed method is evaluated for the network throughput. Section VI shows network simulation results. Finally, we conclude this paper and mention the future work in Section VII.

II. Related Work

In this section, we review existing methods for measuring the throughput of the networks and routing methods for improving the throughput.

A. High-Throughput Large-Scale Networks

Large-scale data center networks generally use Fat-Tree topologies [3], but it is disadvantageous in that there exist many communication hotspots and most of the network is underutilized [7]. One possible solution for eliminating such bottlenecks is to design reconfigurable networks such as optical wireless communications. However, they require high costs, and thus they would not currently be reasonable.

DLN [1] and Jellyfish [2] apply random graphs [8] to the topology of interconnection networks. Slim Fly [9] and Xpander [10] are semi-optimal network topologies for given the number of switches and the number of ports per switch. The four networks above achieve balanced throughput for arbitrary traffic and communication loads. In particular, the

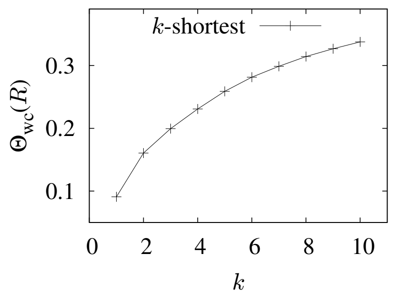


Fig. 1: Worst-case throughput achieved by k-shortest path routing (64 nodes, degree of 4).

network topologies adopting random graphs are promising because they make network sizes and the number of ports flexible. By using a fluid-flow model, Jyothi et al. show that such *random topologies* improve the throughput as compared with Fat-Tree topologies if we use the optimal routing between end nodes [11].

B. Maximizing Throughput by Routing

Wang et al. propose the multi-path routing algorithm called *SCRAT* [12], which increase the bandwidth compared to the conventional multi-path routing. However, SCRAT considers the bandwidth of an independent flow for each source-destination pair and does not consider the multiple concurrent flows among multiple source-destination pairs.

For Jellyfish, the *k-shortest path routing* is proposed to improve the network throughput [2]. Yuan et al. propose the *limited length spread k-shortest path routing* (*LLSKR*), which improves the k-shortest path routing by more effectively exploiting the path diversity [13].

Figure 1 shows the relationship between the throughput and the number k of paths used in the k-shortest path routing with 64 4-degree nodes. We show here the worst-case throughput for all of the possible traffic patterns. Obviously, the throughput increases with the value of k.

The k-shortest path routing and LLSKR improve the communication throughput by reducing the number of occupied channels. On the one hand, longer-than-shortest paths increase the number of occupied channels, and consequently they decrease the throughput, as shown in [13]. On the other hand, however, using longer paths instead of short paths may avoid the bottleneck of specific channels along the short paths and improve the network throughput.

This paper explores the use of longer-than-shortest paths for improving the network throughput. More specifically, we measure the path utilization when the network throughput for all of the traffic matrices are maximized. We propose the method for generating routing paths such that the network throughput is maximized by exploiting longer-than-shortest paths.

III. Optimization of Worst-Case Throughput with Linear Program

Static routing algorithm design can be modeled as a multi-commodity flow problem (MCF). In this model, communication flows on the network so that the load on each channel does not exceed a certain capacity.

In this work, we explore a method for generating multiple paths to improve the network throughput. We use the framework of the existing method [6] to find the worst-case throughput under adversarial traffic. In the same way as this framework, we provide some definitions about a routing method and the network throughput.

A. Routing Algorithm

First, a static routing algorithm defines a set of paths available in the network for each source-destination pair. To describe a routing algorithm R, we let $R(p)$ be the probability that a packet uses the path p. Then, R can represent a valid static routing algorithm as long as the following conditions are satisfied.

$$\begin{array}{ll} \sum_{p\in P_{s,d}} R(p) = 1 & \forall s,d \in N, \\ R(p) \geq 0 & \forall p \in P, \end{array}$$

where P and $P_{s,d}$ represent the set of all paths and the set of paths between source s and destination d, respectively. Each path is a *simple path* that eliminates any loop and any revisit channel. This formulation creates a commodity flow for each of the N^2 source-destination pairs in the network.

B. Metrics for Measuring Network Throughput

Using the multi-commodity flow formulation introduced in the previous section, several network metrics can be defined. By our definition, the maximum throughput in a network can sustain under a given traffic pattern. The throughput is determined by the channel loads. That is, once the average load on a channel reaches the channel's capacity, that channel is saturated. The first channel to saturate becomes a bottleneck and thus determines the network's throughput.

The expected number of packets that cross a particular channel c per cycle, referred to as the load γ_c, is the sum of the loads contributed by each source-destination pair. In terms of the traffic matrix π and the routing algorithm R,

$$\gamma_c(R,\pi) = \sum_{s,d\in N} \pi_{s,d} \sum_{\substack{p:c\in p,\\ p\in P_{s,d}}} R(p),$$

where $\pi_{s,d}$ is a binary value that represents whether communication exists for the (s,d)-pair. In the traffic matrix π, each source node sends packets to exactly one destination node, while each destination node receives packets from exactly one source node. That is,

$$\begin{array}{ll} \pi_{s,d} \in \{0,1\} & \forall s,d \in N, \\ \sum_{d\in N} \pi_{s,d} = 1 & \forall s \in N, \\ \sum_{s\in N} \pi_{s,d} = 1 & \forall d \in N. \end{array}$$

We set the value of each channel's capacity $b_c = 1$, which is the same across all channels in the network. By applying the above formulations to each channel load, we define the maximum channel load across the whole network as

$$\gamma_{\max}(R,\pi) = \max_{c\in C} \left[\gamma_c(R,\pi)\right].$$

This maximum channel load defines the maximum throughput $\Theta(R,\pi)$ of the network under the traffic matrix π as

$$\Theta(R,\pi) = \gamma_{\max}(R,\pi)^{-1}.$$

Each source-destination pair can send and receive packets at up this ratio of each channel's capacity under the traffic matrix π without saturating all channels in the network.

In the above formulations, the channel load γ_c is linear in the routing algorithm R. Moreover, since it is the maximum value in a set of γ_c for all channels, the maximum channel load $\gamma_{\max}$ is convex in R. The existing method [6] uses these properties to establish a linear program that can obtain the worst-case channel load for a routing algorithm R and an arbitrary traffic matrix π.

C. *Optimization of Throughput under Given Traffic π*

The problem of designing a routing algorithm R with the optimal throughput for a traffic matrix π can be expressed as a linear program. The maximum channel load for R and π is defined as $\gamma(R,\pi)$. We introduce a new variable w that maintains $w \geq \gamma(R,\pi)$. The value w is used as an objective function in the following linear program:

$$\begin{array}{lll} \text{minimize} & w & \\ \text{subject to} & \sum_{p\in P_{s,d}} R(p) = 1 & \forall s,d \in N, \\ & R(p) \geq 0 & \forall p \in P, \\ & \gamma_c(R,\pi) \leq w & \forall c \in C. \end{array} \quad (1)$$

D. *Optimization of Throughput under Adversarial Traffic*

We also consider the problem of designing a routing algorithm R with the optimal worst-case throughput under adversarial traffic. In a similar way to Section III-C, the maximum channel load for R under any traffic is defined as $\gamma_{\text{wc}}(R)$. For some channel c and traffic matrix π, $\gamma_c(R,\pi) = \gamma_{\text{wc}}(R)$ is satisfied. This equality induces a new variable w that satisfies $w \geq \gamma_{\text{wc}}(R)$. This variable is used as an objective function in the following linear program:

$$\begin{array}{lll} \text{minimize} & w & \\ \text{subject to} & \sum_{p\in P_{s,d}} R(p) = 1 & \forall s,d \in N, \\ & R(p) \geq 0 & \forall p \in P, \\ & \gamma_c(R,\pi) \leq w & \forall c \in C, \pi \in \Pi, \end{array}$$

where a set of traffic matrices Π is $|N|$-permutations for a set of nodes N. The size of Π becomes $|N|!$, which make the linear program difficult to solve in a practical time.

In order to reduce the number of constraints to a polynomial number, the Lagrange dual function and the Birkhoff–von Neumann theorem can be utilized [6]. We introduce new variables v and u to reformulate the problem. The resulting linear program is:

$$\begin{array}{lll} \text{minimize} & w & \\ \text{subject to} & \sum_{p\in P_{s,d}} R(p) = 1 & \forall s,d \in N, \\ & R(p) \geq 0 & \forall p \in P, \\ & \sum_{\substack{p:c\in p,\\ p\in P_{s,d}}} R(p) \leq v_{d,c} - u_{s,c} & \forall s,d \in N, c \in C, \\ & \sum_{d\in N} v_{d,c} - \sum_{s\in N} u_{s,c} = w & \forall c \in C. \end{array} \quad (2)$$

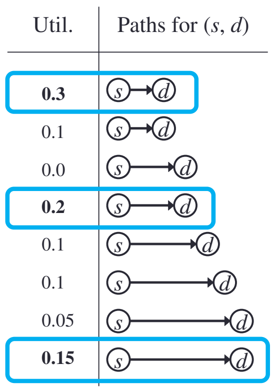


Fig. 2: Example of path selection ($m = 8, k = 3$).

IV. Generating k-Optimized Paths

In this section, we propose k-optimized path routing which can replace the conventional k-shortest path routing. The proposed method uses the linear program shown in the previous section. As in the case of k-shortest path routing, k paths are generated for each source-destination pair. In order to determine the k paths, firstly m-shortest paths are generated, where m is enough larger than k. k paths that can maximize the throughput are then obtained from among the m paths. Using m-shortest paths as inputs, a path search is performed with the linear programming as the following procedures.

(i) For a set of available paths P, the maximum channel load is calculated using the linear program. Depending on whether traffic is given or not, the calculation is performed as follows.

  (A) The traffic-independent method calculates the maximum channel load under adversarial traffic using the linear program (2) in Section III-D.

  (B) The traffic-dependent method calculates the maximum channel load under a given traffic π using the linear program (1) in Section III-C.

(ii) The utilization ratio of each path is extracted in the optimization result. The top k paths for each source-destination pair are then selected from the m paths.

The top k paths may include longer-than-shortest paths that are not in k-shortest paths. These paths can improve the network throughput by using them as substitutes for the shortest paths with low utilization.

Figure 2 shows an example of the path selection. The step (i) obtains the channel loads when using m-shortest paths for each source-destination pair. At the same time, the utilization ratio of each path is calculated in the optimization. The step (ii) extracts detour paths that can avoid bottlenecks. These paths are included in k available paths to generate routing optimized for the network throughput.

V. Evaluation

The proposed k-optimized path routing introduced in Section IV is compared with the conventional k-shortest path routing. Gurobi [14] is used as a linear program solver. A barrier method is used for the optimization method in the linear program. Default values of parameters are as follows. The number of paths to be searched is set to $m = 10$. The number of paths finally used in routing is set to $k = 3$.

A. Evaluation Metrics

The proposed and conventional methods are evaluated for the following two metrics.

(a) Network throughput is calculated with the linear programs shown in Section III.

(b) Average path length for each source-destination pair achieved by routing is evaluated. We define the path length of each source-destination pair as the summation of weighted lengths of the available paths. We weight the length of each path p with the utilization rate $R(p)$ extracted from the optimization result. That is, the average path length $H_{\mathrm{avg}}(R)$ is defined as

$$H_{\mathrm{avg}}(R) = \frac{\sum_{\substack{s \neq d \\ s,d \in N}} \sum_{p \in P_{s,d}} R(p) \cdot h(p)}{|N| \cdot (|N| - 1)},$$

where the length of the path p is represented as $h(p)$.

B. Traffic-Independent k-Optimized Path Routing

We evaluate the proposed k-optimized path routing that is implemented based on the traffic-independent method (A) shown in Section IV. The proposed and conventional methods are applied to random regular graphs that have the same degree of each node. The network size and degree are set to 64 and 8, respectively.

Figure 3 shows comparisons between the proposed and conventional methods varying the parameter k (the number of available paths in routing) with the fixed parameter $m = 10$. Figure 3a shows that the proposed k-optimized path routing achieves higher worst-case throughput than the conventional k-shortest path routing. It can improve the throughput under adversarial traffic by up to 17.5 %.

As described in Section IV, k-optimized path routing utilizes longer-than-shortest detour paths to avoid bottlenecks. These paths degrade the average path length, which can be seen in Figure 3b. In this evaluation, the maximum rate of the degradation is 3.4 %. Considering the results in Figure 3a, k-optimized path routing modestly degrades the average path length while drastically improves the network throughput by utilizing longer-than-shortest detour paths.

Figure 4 shows comparisons between the proposed and conventional methods varying the parameter m (the number of paths to be searched) with the fixed parameter $k = 3$. When increasing the number of paths to be searched m, ideally the worst-case throughput is also increased. However, as shown in Figure 4a, the value of the worst-case throughput becomes the maximum when $m = 7$. The rate of the improvement compared with the k-shortest path routing is 29.5 %.

When $m > 7$, the improvement rate is degraded to 20.7 % as the value of m increases. This is because the proposed method uses a simple heuristic that chooses paths with the high utilization ratio obtained in the optimization result. The paths used with high probability in m-shortest path routing are not always the optimal paths to be selected as the k paths for the maximum worst-case throughput. Improving the heuristic for selecting k-optimized paths is one of our future works.

As shown in Figure 4b, the average path length is increased by up to 3.5 % due to detour paths. However, at $m = 7$ where the worst-case throughput becomes the maximum, its degradation rate remains at 2.1 %. Therefore, one of our future works is also developing a method of searching for the value m that can maximize the throughput while suppressing the degradation of the average path length.

C. Traffic-Dependent k-Optimized Path Routing

We evaluate the proposed k-optimized path routing that is implemented based on the traffic-dependent method (B) shown in Section IV. As in Section V-B, the proposed and conventional methods are applied to random regular graphs. The network size and degree are set to 256 and 16, respectively. Longest Matching traffic (LM) [11] is used as given traffic.

Figure 5 shows comparisons between the proposed and conventional methods varying the parameter k with the fixed parameter $m = 10$. Figure 5a shows that the k-optimized path routing achieves higher throughput for LM traffic. It can improve the throughput by up to 69.2 %.

Unlike the traffic-independent method shown in Section V-B, the traffic-dependent method hardly degrades the average path length with detour paths, as shown in Fig 5b. The maximum rate of the degradation is 0.0016 %. Hence the traffic-dependent k-optimized path routing can improve the throughput while maintaining the low network latency.

Figure 6 shows comparisons between the proposed and conventional methods varying the parameter m with the fixed parameter $k = 3$. Figure 6a shows that the value of the throughput becomes the maximum when $m = 22$. The rate of the improvement compared with the k-shortest path routing is 133 %. For the same reason as the traffic-independent method shown in Section V-B, the traffic-dependent method cannot sustain the maximum throughput when $m > 22$.

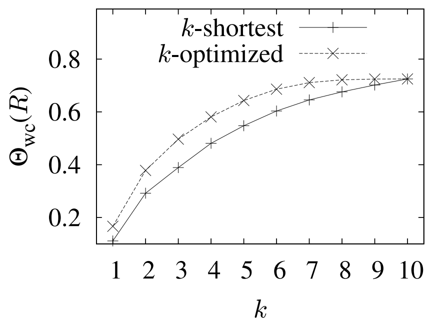
Figure 6b shows that the average path length is increased by up to 0.02 % due to detour paths. Similarly to Fig 5b, the degradation rate is much smaller than that of the traffic-independent method.

VI. Network Simulation

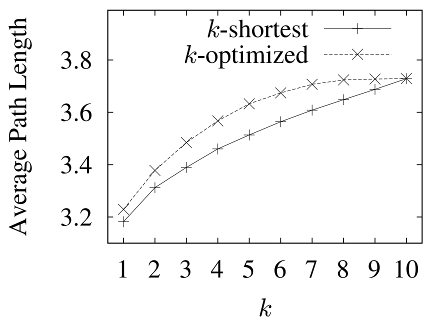
A cycle-accurate network simulator Booksim [17] is used for evaluation. Network parameters for the simulation are shown in Tab. I. In order to avoid deadlocks, Duato's protocol [15] is used. In this protocol, we adopt the up*/down*

TABLE I: Network parameters.

Simulation period	10,000 cycles
Packet size	1 flit
Number of VCs	2
Buffer size per VC	8 flits
Number of pipeline stages	4
Escape path [15]	up*/down* routing [16]

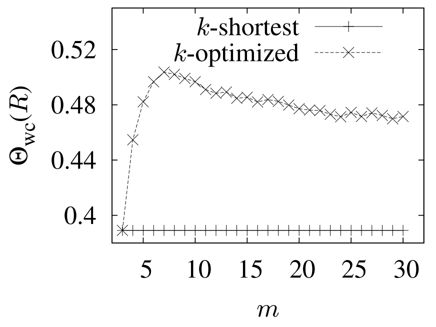


(a) Worst-case throughput.

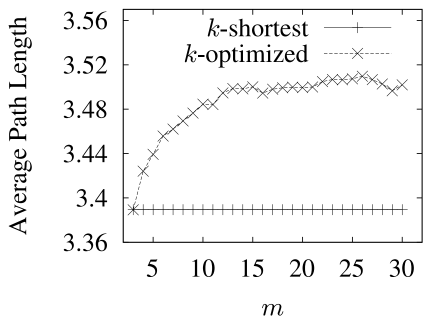


(b) Average path length.

Fig. 3: Traffic-independent k-optimized path routing for 64 nodes with different k ($m = 10$).

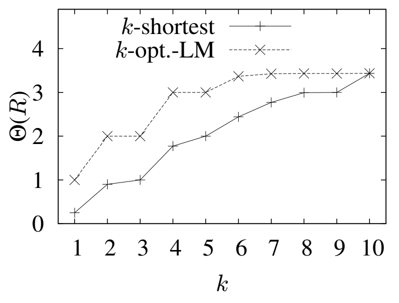


(a) Worst-case throughput.

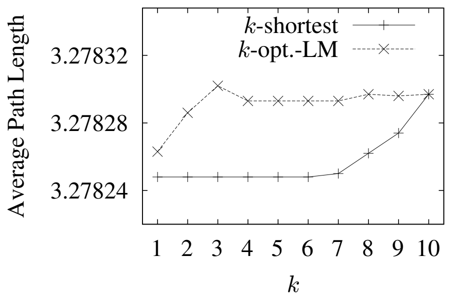


(b) Average path length.

Fig. 4: Traffic-independent k-optimized path routing for 64 nodes with different m ($k = 3$).

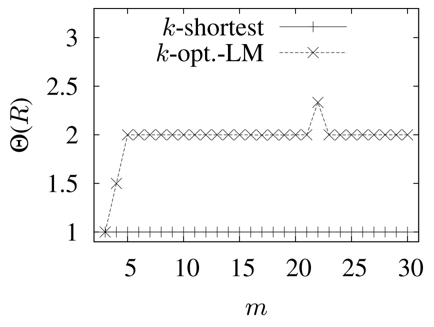


(a) Throughput.

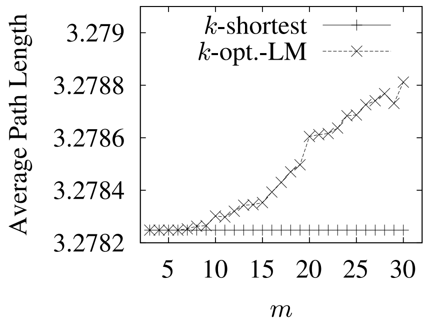


(b) Average path length.

Fig. 5: Traffic-dependent k-optimized path routing for 256 nodes with different k ($m = 10$).



(a) Throughput.



(b) Average path length.

Fig. 6: Traffic-dependent k-optimized path routing for 256 nodes with different m ($k = 3$).

routing [16] with the spanning trees optimization method [18] as the escape paths.

In this evaluation, the proposed traffic-dependent and traffic-independent k-optimized path routing methods are compared with the conventional k-shortest path routing. The parameters are set to $m = 10$ and $k = 3$. The evaluated network and its size and degree for each method are the same as in Section V-B and V-C. For the traffic-independent method, we use four kinds of traffic: uniform, transpose, shuffle, and reverse. For the traffic-dependent method, we use the following traffic: bit complement, transpose, shuffle, and reverse.

Figure 7 shows the network performance of the traffic-independent k-shortest path routing. As seen in this figure, regardless of adopted traffic, the network bandwidth is improved without optimizing for the traffic. The maximum improvement rate is 14 %. This rate is smaller than the improvement rate of the worst-case throughput shown in Figure 3a. This is because Duato's protocol induces another deadlock-free network layer for up*/down* routing. That is, the result values of the bandwidth are raised by that of the deadlock-free network. It is also notable that k-optimized path routing achieves almost the same network latency at the low accepted flit rate as the conventional k-shortest path routing.

Figure 8 shows the network performance of the traffic-dependent k-shortest path routing. As seen in this figure, the network bandwidth is improved by optimizing for each given traffic. The maximum improvement rate is 16 %. Moreover, similarly to the traffic-independent method, it can achieve almost the same network latency as the k-shortest path routing.

VII. Conclusion and Future Work

In this work, we propose k-optimized path routing which utilizes k paths for each source-destination pair to improve the network throughput for low-latency random networks.

Conventional k-shortest path routing for random networks improves the network throughput by increasing the number of available paths k for each source-destination pair. However, this method may decrease the network throughput when k is small because it cannot use detour paths that would avoid bottlenecks to improve the throughput.

In the proposed k-optimized path routing, a linear program is exploited to calculate the utilization rate of m-shortest paths when the throughput is maximized. Furthermore, the top k paths with high utilization rate are used in the proposed routing to maximize the network throughput for the limited number of available paths k.

We implement both traffic-independent and traffic-dependent k-optimized path routing. The traffic-independent method can improve the worst-case network throughput under adversarial traffic by up to 29.5 %. The traffic-dependent method can also improve the throughput under traffic with the longest paths by up to 133 %. Moreover, both of the proposed methods can improve the network bandwidth while maintaining the low network latency.

As future work, we will focus on improving the heuristic for selecting k-optimized paths. We will also focus on developing a method of searching for the value m that can maximize the throughput while suppressing the degradation of the average path length.

References

[1] 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.

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

[3] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transactions on Computers (TC)*, vol. 34, no. 10, pp. 892–901, Oct 1985.

[4] C. Hopps, "RFC 2992 – Analysis of an Equal-Cost Multi-Path Algorithm," Internet Engineering Task Force, Tech. Rep., 2000.

[5] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science, Theory Series*, vol. 17, no. 11, pp. 712–716, Jul 1971.

[6] B. Towles, W. J. Dally, and S. Boyd, "Throughput-Centric Routing Algorithm Design," in *Proc. of the 15th annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, Jun 2003, pp. 200–209.

[7] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, Aug 2017, pp. 281–294.

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

[9] 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.

[10] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards Optimal-Performance Datacenters," in *Proc. of the 12th International on Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Dec 2016, pp. 205–219.

[11] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla, "Measuring and Understanding Throughput of Network Topologies," in *Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2016, pp. 761–772.

[12] H. Wang, K. Qian, C. Hu, C. Zhang, and Y. Zhou, "Routing Optimization for Server-Centric Data Center Networks," *Computer Science and Information Systems*, vol. 13, no. 2, pp. 593–608, Jun 2016.

[13] X. Yuan, S. Mahapatra, W. Nienaber, S. Pakin, and M. Lang, "A New Routing Scheme for Jellyfish and its Performance with HPC Workloads," in *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2013, pp. 1–11.

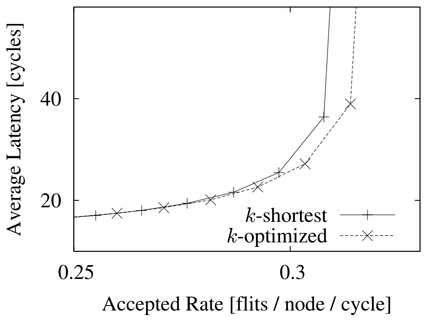
[14] "Gurobi Optimization Inc. Gurobi optimizer reference manual." http://www.gurobi.com.

[15] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," in *Proc. of the International Conference on High-Performance Computing (HiPC)*, Dec 1997, pp. 330–335.

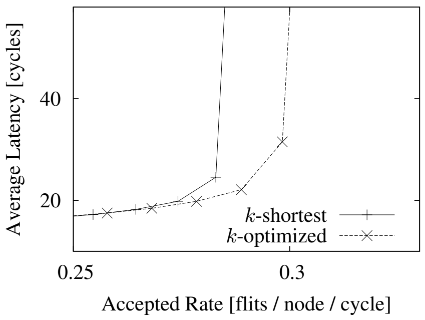
[16] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, and T. L. Rodeheffer, "Autonet: A High-speed, Self-configuring Local Area Network Using Point-to-point Links," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1318–1335, Oct 1991.

[17] 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.

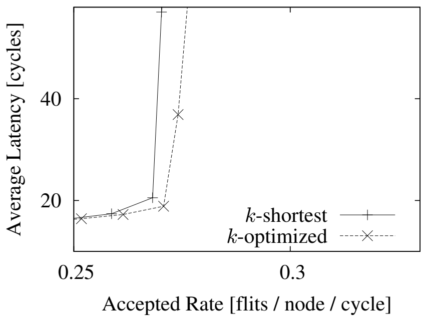
[18] H. Matsutani, P. Bogdan, R. Marculescu, Y. Take, D. Sasaki, H. Zhang, M. Koibuchi, T. Kuroda, and H. Amano, "A Case for Wireless 3D NoCs for CMPs," in *Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2013, pp. 22–28.



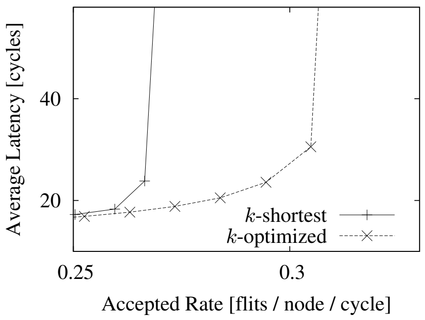
(a) Uniform traffic.



(b) Transpose traffic.

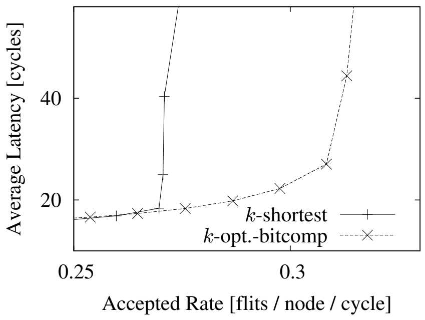


(c) Shuffle traffic.

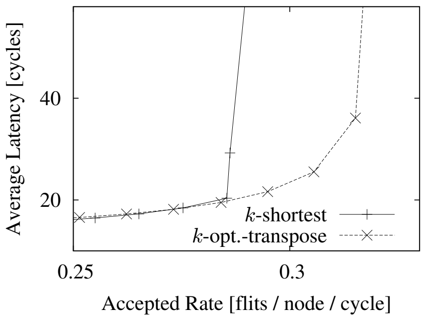


(d) Reverse traffic.

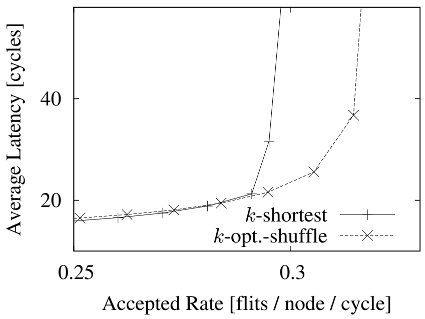
Fig. 7: Network performance of traffic-independent k-optimized path routing for 64 nodes.



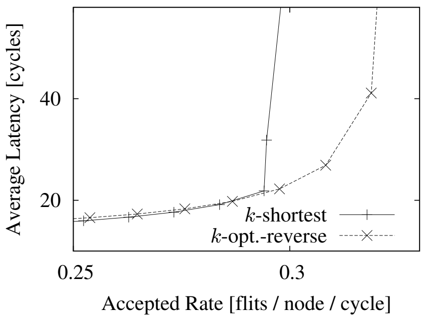
(a) Bit complement traffic.



(b) Transpose traffic.



(c) Shuffle traffic.



(d) Reverse traffic.

Fig. 8: Network performance of traffic-dependent k-optimized path routing for 256 nodes.