Tsinghua Science and Technology

Volume 24 | Issue 4

Article 1

2019

TIO: A VLC-Enabled Hybrid Data Center Network Architecture

Yudong Qin
the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.

Deke Guo
the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.

Guoming Tang
the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.

Bangbang Ren
the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.

Recommended Citation

Yudong Qin, Deke Guo, Guoming Tang et al. TIO: A VLC-Enabled Hybrid Data Center Network Architecture. Tsinghua Science and Technology 2019, 24(04): 484-496.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/11 pp484–496
DOI: 10.26599/TST.2018.9010093
Volume 24, Number 4, August 2019

TIO: A VLC-Enabled Hybrid Data Center Network Architecture

Yudong Qin, Deke Guo*, Guoming Tang, and Bangbang Ren

Abstract: To satisfy the ever-increasing bandwidth demand of modern data centers, researchers have proposed *hybrid* Data Center Networks (DCNs), which employ high-bandwidth Optical Circuit Switches (OCSs) to compensate for Electrical Packet Switches (EPS). Existing designs, such as Helios and c-Through, mainly focus on reconfiguring optical devices to meet the estimated traffic requirements. However, these designs face two major challenges in their OCS-based networks, namely, the complex control mechanism and cabling problems. To solve these challenges, we propose TIO, a hybrid DCN that employs Visible Light Communication (VLC) instead of wired OCS design to connect racks. TIO integrates the wireless VLC-based Jellyfish and wired EPS-based Fat Tree seamlessly and combines the opposite and complementary characteristics, including wireless VLC direct connection and wired electrical packet switching, random graph, and Clos topology properties. To further exploit the merits of TIO, we design a hybrid routing scheme and congestion-aware flow scheduling method. Comprehensive evaluations indicate that TIO outperforms the Jellyfish and Fat Tree in both topology properties and network performance, and the flow scheduling method also evidently improves performance.

Key words: Data Center Network (DCN); visible light communication; wireless links; topology design

1 Introduction

Data centers are dominating infrastructures that support various cloud computing applications[1]. To satisfy bandwidth demand, modern data centers depend heavily on fiber-optic links[2]. In this regard, researchers have proposed many *hybrid* Data Center Network (DCN) structures, which utilize Optical Circuit Switches (OCSs) to compensate for existing Electrical Packet Switches (EPS), such as Helios[3] and c-Through[4]. Unlike EPS, OCSs simply redirect light from one port to another, which costs less power, supports high link bandwidth, and avoids expensive optical-electrical-optical signal conversion[5]. However, two major challenges still exist for wired OCS-based networks in most existing hybrid designs.

The first challenge is the complex control mechanism. Current proposals that employ OCS mainly focus on flexibility. In these designs, the optical circuits in data centers can be reconfigured according to the current or predicted traffic demands. However, to achieve such reconfigurations, data centers require the ability to maintain strict time synchronization among network devices, to estimate the network-scale traffic demand, and to assign circuit configurations in a centralized manner[5]. All these issues make the control plane complex and unwieldy. In addition, the reconfiguration processes in the existing designs cause high latency. The link setup and switching time of optical components are far from their theoretical limits[6]. Thus, in this paper, we propose a DCN that differs from the prevalent approach by not dynamically reconfiguring optical links in response to traffic estimation. Hence, it requires no demand collection, no switch scheduling algorithm, and no network-scale synchronization.

• Yudong Qin, Deke Guo, Guoming Tang, and Bangbang Ren are with the Science and Technology on Information System Engineering Laboratory, National University of Defense Technology, Changsha 410073, China. E-mail: qinyudong12 @nudt.edu.cn; guodeke@gmail.com; gmtang@nudt.edu.cn; renbangbang94@163.com.

∗ To whom correspondence should be addressed.

Manuscript received: 2018-01-27; revised: 2018-03-10; accepted: 2018-03-14

The second challenge is the cabling complexity of wired OCS, which is inherently difficult and error-prone. In these networks, a large number of optical cables result in design and development problems related to wire ducting and maintenance, heat dissipation, and operation costs (about 7%–8% of the total infrastructure cost)[7]. For a large-scale data center, connecting a large number of servers and switches usually takes considerable manual effort and time. Massive cables also pose difficulties to maintenance, troubleshooting, and reconfiguration. Furthermore, the large volume of space occupied by cable bundles restricts air flow, resulting in inefficient cooling and increased energy consumption.

To solve the cabling problem in wired OCS-based networks of hybrid DCNs, researchers propose replacing optical cables with wireless links[8] and offer diverse wireless OCS-based networks that employ optical wireless communication, which is also known as Free Space Optics (FSO). In FSO communication, a light transmitter is simply a single-mode fiber connected to a beam collimator and transmits a modulated optical beam from the signal source[9]. Most existing wireless OCS-based designs, such as Firefly[10] and ProjecToR[11], use laser as wireless FSO links to connect racks. In this paper, we note that the emerging Visible Light Communication (VLC) technology can be used to construct a wireless data center because its data rate and transmission range can satisfy the demand of DCN. VLC has its own advantages over the widely used laser. First, the VLC transceivers are lower-cost and less-complicated, because VLC uses Light-Emitting Diodes (LEDs) to generate signals. Second, VLC is more safe for humans, whereas laser may harm the eyes and the skin. Third, laser is difficult to apply for large-scale commercial use. Thus, we believe that VLC is a promising alternative and can be used to construct a wireless network in a hybrid DCN.

In this paper, we envision the following three design rationales to construct high-performance VLC-based hybrid DCNs: (1) **Wireless**. It uses wireless communication technology (i.e., VLC) to achieve stable connection among racks. (2) **Plug-and-play**. It does not need a centralized control mechanism, and it does not reconfigure wireless links to match traffic conditions. (3) **Easily deployable**. It needs no additional infrastructure-level addition or adjustment within data centers. The above rationales can bring profound benefits to data centers. First, hierarchical switches and inter-rack cables are no longer needed with the construction of a wireless data center, that is, hardware investment can be reasonably reduced, and it also benefits maintenance and cooling. Second, given that a centralized control mechanism is not needed, no demand collection, no switch scheduling algorithm, and no network-scale synchronization are necessary, thereby simplifying the work progress. Third, the use of a VLC does not require altering the physical infrastructures in data centers, for example, changing the ceiling into a mirror. Thus, it is easy to install and maintain.

However, existing proposals cannot achieve all three design rationales simultaneously. First, these designs mainly focus on flexible reconfiguration of links. Thus, they implement a centralized control mechanism and impose frequent and complicated manipulation on optical devices. Moreover, infrastructure-level alteration to the environment of data centers is needed to achieve wireless connection. For example, Firefly needs to install a reflective ceiling to relay optical signals. ProjecToR requires the installation of a micro-mirror array and mirror assembly for each signal launcher, which is hard to deploy and maintain.

In this paper, we propose TIO to achieve the above three rationales simultaneously. To form a wireless network, VLC links are employed in TIO to transmit data among racks. Inspired by the good performance of random graphs that are proven to have optimal throughput for uniform traffic[12], we use VLC links to construct a wireless Jellyfish[13]. To eliminate complex control on optical devices and provide stable connection, the VLC links are set as static links. In this case, to efficiently deal with skewed traffic, we rely on the previously demonstrated approach to construct a hybrid network with EPS[3,5]. Specifically, we integrate the wireless Jellyfish with the wired Fat Tree, a representative instance of Clos network. Thus, TIO is a hybrid network structure. It combines together the advantages of wireless VLC direct connection and wired electrical packet switching, random graph, and Clos topology properties. Hence the proposed network structure is called *TIO*, a *two-in-one* data center network architecture.

The major contributions of this paper are summarized as follows:

- We design TIO, a hybrid data center network that integrates the wireless Jellyfish and wired Fat Tree thereby possessing the merits of wireless VLC direct connection and wired electrical packet switching, random graph, and Clos topology properties. It is a hybrid DCN with a wireless, plug-and-play, and easily deployable VLC-based network.

• To fully exploit the benefits of the topology of TIO, we design a hybrid routing scheme that allows it to jointly utilize both wireless and wired links. To further improve the network performance, we propose a congestion-aware scheduling method to optimize the flow scheduling problem in the network.

• Trace-driven experiments are conducted to evaluate the performance of TIO under different traffic patterns. Results show that our TIO can achieve enhanced performance, as expected.

The remainder of this paper is organized as follows. Section 2 summarizes prior designs of data centers and introduces the background of VLC. Section 3 reports how the hybrid structure of TIO is constructed. Section 4 presents the hybrid routing method for TIO, and proposes the congestion aware flow scheduling method. Section 5 evaluates the performance of TIO in both topological properties and network performance. Section 6 concludes this paper.

2 Related Work and Background

In this section, related works about DCNs employing OCS are reviewed. Then, we introduce VLC technology in brief and discuss the feasibility of employing VLC links to support data transmission in data centers.

2.1 OCS-based data center networks

To meet the ever-increasing bandwidth demand, high-bandwidth OCSs are widely used in the design of DCNs. Compared with EPS, OCS supports far larger bandwidth to satisfy the demand of data-intensive applications. Researchers have proposed many DCN designs that employ OCS to construct data center networks. The physical layer employed by these designs contains fiber-coupled optical crossbar switches, wavelength division multiplexing switches, and free-space optical transceivers, among others[5].

Helios[3] and c-Through[4] independently propose two network architectures to achieve high bandwidth and low-latency communication among massive racks. They both construct a hybrid network that combines the merits of electrical packet switches and optical circuit switches[6]. On the basis of the predicted traffic demand, optical network connects each rack to exact one other rack. As a result, pairs of racks experience transient high capacity links, and the rack pairs change over time. In these networks, the circuit-switched portion handles the baseline, slowly changing interpod communication, while the packet-switched delivers all-to-all bandwidth for bursty traffic. Thus, the strengths of one kind of switch compensate for the weaknesses of the other kind. However, these designs suffer from constraint rack-level connectivity[14], and the reconfiguration of circuits can incur a long switching delay.

Some wireless designs employ FSO links to eliminate wiring. In these designs, the hierarchical switches and inter-rack cables are no longer needed, thereby reducing the hardware investment. Firefly[10] and ProjecToR[11] leverage lasers to achieve fully wireless networks on the rack level, and all links are reconfigurable. The difference between them is that Firefly adopts the centralized control plane, whereas the control plane of ProjecToR is distributed. Firefly reconfigures the whole network based on the estimated traffic demand to maximize throughput. In ProjecToR, each switch independently reconfigures laser beams based on only the local racks' demand. Unfortunately, these designs are difficult to deploy and require considerable alterations to the environment of data centers, as we mentioned above. Moreover, the reliance on FSO links causes many other practical concerns (e.g., robustness to dust and vibration)[14].

However, all the above designs need a powerful but complex control mechanism to support reconfiguration based on the estimated demand. Some recent proposals attempted to decouple the reconfiguration from traffic demand. Flat-tree[15] decouples the two slightly by not strictly matching topologies with demand. It has three preset configuration modes, and provides the most suitable one for a service. RotorNet[5] has a series of pre-determined switch port matchings. However, it shifts from these matchings periodically without any dynamic optimization in response to traffic estimation. MegaSwitch[14] is a static network that uses wavelength division multiplexing. Racks are arranged in an optical ring, and senders broadcast traffic on this fiber ring. The control mechanism needs to assign a wavelength to senders and ensure that no same wavelengths reach the same receivers. The scale of MegaSwitch is limited to 33 racks by the current hardware.

In conclusion, most existing designs cannot simultaneously match our three proposed design rationales in their OCS-based network, because they are more interested in flexibility. In this paper, we propose TIO to achieve all three rationales and show its unique merits.

2.2 Feasibility of employing VLC links in DCNs

We first introduce the working principle of VLC briefly and then analyze the feasibility of interconnecting racks by using VLC links.

VLC is being developed rapidly. For VLC, data transmission is achieved through intensity modulation of visible spectrum LEDs or laser diodes. It uses the on-off keying modulation scheme, where "1" and "0" are represented by the presence or absence of light, respectively[16]. It is the simplest form of digital communication. In practice, few data centers employ VLC links. The VLC is not yet commercialized. However, VLC has many advantages in communication, thereby making it feasible for introduction into DCNs.

High data rate. Under the control of microchips, LEDs can complete switching on and off millions of times in a second. Researchers already achieved a 10 Gbps data rate in 2013. Hence, VLC links are believed to be capable of transmitting data in a data center.

No interference. Other wireless communication methods, for example, 60 GHz, may generate electromagnetic interference in the communication process. This condition means that the communication signals may interfere with each other. By contrast, no radiation and interference occur among VLC signals, which makes it suitable for serving intensive links in a data center.

Sufficient transmission range. For VLC, long distance transmission results in limited data rate and high energy cost. Fortunately, LED-based VLC links can reach data rate of 10 Gbps with 10 m. Laser-based transceivers can achieve fast long-distance communication (in the order of kilometers) with a high data rate, due to its outstanding directionality.

High security. Visible light cannot penetrate opaque objects. Thus, information leakage from severer rooms is not a consideration. However, this situation also has line-of-sight limitation. Thus, to use VLC in a data center, light paths should be designed carefully and ensure that they are not blocked by any infrastructure component.

The successful application of VLCcube[17] is a strong evidence of the feasibility. VLCcube was the first to introduce VLC links for transmission in data centers. It constructs a wireless Torus with VLC links, which is used to enhance the performance of wired links.

3 Design of TIO

In this section, we illustrate the construction of TIO topology, a hybrid structure that integrates the wireless Jellyfish and wired Fat Tree seamlessly.

3.1 Structure of TIO

We proposed three design rationales for OCS-based networks: wireless, plug-and-play, and easily deployable. To achieve wireless and stable connection, we employ VLC links to build wireless VLC-based networks. To achieve plug-and-play, each VLC links are set to be static, thereby eliminating complex control on optical circuit switches. For easy deployment, mirrors and other devices that are used to relay light signals are not adopted.

A recent analysis of Facebook traffic reveals that communication inside a data center is extremely wide-spread[18]. Compared with Clos topologies, random graphs provide better support for network-wide traffic. Random graph is a perfect match for uniform global traffic[15]. On the basis of these considerations, we utilize VLC links to connect ToR switches as the topology of Jellyfish, a random regular graph. This design brings an added benefit, that is, the employment of wireless links eliminates cabling problems, one of the main challenges in the construction of Jellyfish[13].

However, the traffic in data centers can be unbalanced and generate hot regions. To accommodate bursty and skewed traffic, a typical solution is to maintain a parallel packet-switched fabric[14]. Thus we construct a wired Clos network that has complementary advantages with random graphs. Clos has rich intra-rack bandwidth and is thus suitable for traffic with strong rack-level locality. Specifically, we use electrical packet switches to connect racks as a Fat Tree, because it has full bisection bandwidth.

The size of the wired Fat Tree is related to the port number of switches. Suppose that each switch has k ports. Then the network can be classified into k pods, each of which has $k/2$ ToR switches and $k/2$ aggregate switches. Thus, $k^2/2$ ToR switches are contained, corresponding to $k^2/2$ racks. The wireless Jellyfish is constructed in the space above the racks to avoid blockage. Specifically, we install VLC transceivers on top of the ToR switches. On each ToR switch, we install four transceivers, considering that the upper surface of rack is not large enough.

TIO is a hybrid DCN that integrates the wireless Jellyfish and the wired Fat Tree. We also call this hybrid DCN "two-in-one DCN" because it combines opposite and complementary characteristics, i.e., the wireless VLC direct connection and wired electrical packet switching, random graph, and Clos topology properties. Figure 1 illustrates the topology of the complete TIO, where the port number $k=4$. Figure 2 illustrates the structure of the wireless Jellyfish from the top view. To make the picture concise and graph, we omit half of the VLC links in Fig. 2.

3.2 Joint optimization of wireless Jellyfish and wired Fat Tree

The Jellyfish is a random regular network that provides the desired bisection bandwidth and path length distribution. In TIO, each rack connects to four other racks with wireless VLC links. When constructing Jellyfish, all links are selected randomly, and the constructed Jellyfish is a random 4-regular graph. However, given that TIO is a hybrid network of wireless Jellyfish and wired Fat Tree, we can optimize the topology of Jellyfish incorporated with the properties of Fat Tree.

The path length between any pair of ToRs in Fat Tree

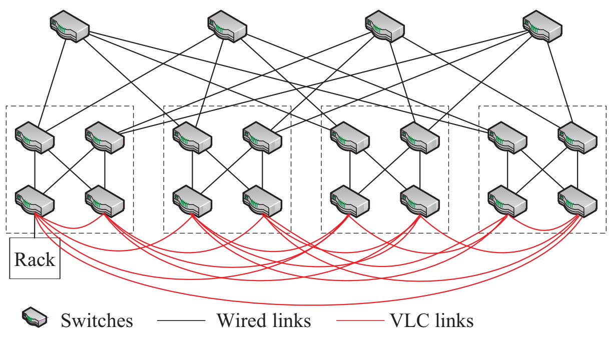


Fig. 1 Example of TIO when $k = 4$.

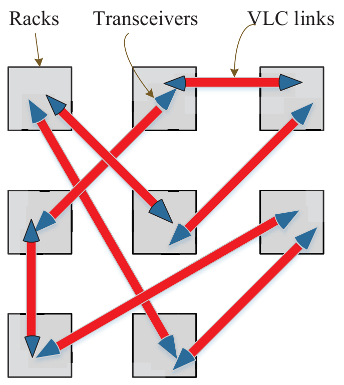


Fig. 2 Top view of racks.

is either two or four hops. Specifically, when two racks are located in the same pod, the path length between them is two hops, and the path length is four hops when the racks belong to different pods. Hence, to shorten the average path length, VLC links should be used to connect two racks in different pods; this approach is better than connecting two racks inside a same pod.

For demonstration purposes, we first assign an identifier to each rack, which consists of two parts. The prefix denotes the pod number, which ranges from 0 to $k-1$. The suffix denotes the rack number in a pod, which ranges from 0 to $k/2-1$. For example, the identifier 43 refers to the fourth racks in the fifth pod.

Then, we express the network in the pod level. In the pod-level graph, each node represents a pod, and the edge between two nodes means that the racks in these two corresponding pods are connected by VLC links directly. Typically, we measure the connectivity of the pod level graph by counting the number of links. Figure 3 depicts an example of the wireless Jellyfish in TIO, with $k=6$. Figure 3a is the Jellyfish in the rack level, and Fig. 3b is in the pod level. The pod-level graph has 6 nodes and 15 links; thus, its connectivity is 15. For the given k, the connectivity of the pod-level graph is no more than $k\times(k-1)/2$.

To make the connectivity in the pod level graph of Jellyfish as large as possible, we generate Jellyfish for many tiems, and select the one with the maximum connectivity. According to the selected Jellyfish topology, we deploy the wireless VLC links above the wired Fat Tree. Through the construction process, we can see that TIO is easily deployable because Fat Tree can be easily implemented[15], and the deployment of wireless Jellyfish is also simple. Moreover, the deployment of the two networks is independent of each other.

3.3 Spatial layout of wireless links

In Jellyfish, links are distributed randomly in the whole

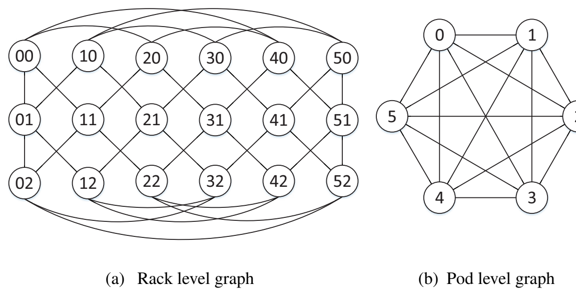


Fig. 3 Topology of TIO in the rack level and the pod level.

network. On the one hand, this property ensures that the network can be constructed at an arbitrary size, and makes the network resilient to failures and miswirings. On the other hand, it further amplifies the cabling problem, because cabling expenditure depends more on the "spread" of cables than on the number or length[19]. In TIO, the employment of VLC links avoids the cabling trouble, while still maintaining good scalability and resilience to failures and miswirings.

However, the introduction of VLC links causes other problems. First, visible light has line-of-sight limitations. Any blockage in the light path will interrupt communication. Thus, a crucial step is to ensure that the VLC links are not blocked by other devices in the data center. Second, the VLC link is not an "ideal thin line" similar to a laser link. The beam width may cause undesired interference to other VLC transceivers because transceivers cannot distinguish the light signals they receive. In addition, given the high sensitivity of VLC transceivers[20], interferences caused by VLC links must be avoided.

To solve the blockage and interference problem, the transceivers are installed at different heights to separate VLC links, which may influence others. For this purpose, we transform this problem into a vertex coloring problem. Typically, this problem requires "colors" to be assigned to all vertices of a graph, and the two vertices of any edge in this graph should be assigned with different colors. Let l represent the number of colors to satisfy this problem. To handle the spatial layout of VLC links (i.e., the installation position above racks) based on the vertex coloring problem, *interference graph* G[21] needs to be computed first to describe the position relationships among all VLC links. Each vertex in G corresponds to a VLC link in Jellyfish, and two vertices in G will be linked only if the original two VLC links intersect in Jellyfish.

Then, we assign different "colors" to vertices in G based on vertex coloring theory, which has been studied for many years. The solution will derive the minimal l. Thus, the VLC links can be divided into l groups according to the assigned colors. The links in the same group have the same assigned color, and are installed at the same height. This method is easy to employ. However, if the transceivers are flexible, then we can decrease the value of l by horizontally adjusting the transceivers' location to avoid some intersections. The effect of this adjustment is fully depended on operators' intelligence, thereby making the adjustment a challenging and interesting task.

4 Routing and Congestion-Aware Flow Scheduling in TIO

Three kinds of routing paths exist between a pair of racks in TIO: wireless, wired, and hybrid paths. The routing algorithms for wireless and wired paths have been studied in previous literature[13,22]. Thus, in this section, we mainly focus on finding hybrid routing paths between rack pairs. To minimize network congestion and balance traffic load, we formulate a congestion-aware flow scheduling model and design scheduling algorithm for the batched traffic pattern.

4.1 Routing scheme in TIO

The TIO has great path diversity. In TIO, each flow can choose to transmit along a wireless, wired, or hybrid path. Consider that each server is connected to one and only one ToR switch. No path diversion exists between servers and the connected ToR switches. Hence, we discuss the routing on the switch level.

4.1.1 Wireless and wired path routing method

For the Fat Tree network, $k^2/4$ shortest paths at length 4 exist between two ToR switches in different pods. Thus, a flow has flexible choices for its route. However, for the Jellyfish network, the number of shortest paths between two ToR switches may be only one. To make full use of the uniformly distributed random links, we use the n-shortest-path routing method in Jellyfish, that is, we derive n shortest paths for every ToR switch pairs. In TIO, we aim to use both wireless and wired links to the same extent. Therefore, we set $n = k^2/4$ to provide ToR pairs the same number choices of wireless paths as wired paths.

Note that the wired and wireless networks do not work separately. In TIO, the wireless Jellyfish integrates tightly with the wired Fat Tree. Aside from fully wireless or wired routing paths, hybrid paths that coexist of both wireless links and wired links between any pair of racks can also be found. In hybrid paths, wireless and wired links can cooperate with each other, thereby enhancing the diversities of routing paths. Thus, we also need a way to derive hybrid paths.

4.1.2 Hybrid path routing method

For any pair of ToR switches, diverse combinations of wireless and wired links exist. Employing the Dijkstra algorithm to calculate the shortest path will incur high time complexity and is impractical, because its computation complexity is $O(k^2)$, and the resultant time consumption will not be acceptable with the increase in k. In this condition, we prefer to search only the hybrid paths, which can shorten the wired path between a pair of ToR switches. Hence, unnecessary computation will be avoided.

For two ToR switches in different pods, the length of the wired path is 4. The length can be shortened by a hybrid path when a VLC link connects the two pods, and one of these two ToR switches is the end of this VLC link. Thus, the deduced hybrid path will be three hops. However, for two ToR switches in the same pod, the length of the wired path is 2, and it cannot be shortened by the hybrid path. In addition, if the VLC link connects the two ToR switches directly, then driving a hybrid path is unnecessary.

On the basis of the above analysis, we propose a routing scheme that consists of the following two steps. Let there be a pair of ToR switches *xy* and *uv* (the identifier of a ToR switch is identical to its corresponding rack). First, given that each ToR switch launches four VLC links, we search the eight VLC links on both *xy* and *uv* to judge whether a link connects pods x and u. If not, then the routing scheme stops here. Second, without loss of generality, we assume that a VLC link (*xy*, *uw*) exist. To complete a hybrid path, an aggregation switch is needed to relay *uw* to *uv*. Given that the ToR switches and aggregation switches in the same pod form a complete bipartite graph, the aggregation switch can be selected randomly. Hence a reasonable hybrid path will be derived.

For example, for two ToR switches 00 and 10 in Fig. 3a, a VLC link connects pods 0 and 1 from 00 to 11. Thus, the length of the hybrid path from 00 to 10 is 3. Compared with the wired path at length 4, the hybrid path does not go through core switch.

The time complexity of this hybrid routing scheme is $O(1)$. The first step needs to search eight VLC links and to judge whether they connect the two pods. The second step selects an aggregation randomly, and it can also be finished at a constant time. Thus this scheme is efficient.

4.2 Problem formulation of flow scheduling

To construct TIO, we introduce wireless VLC links to augment the existing wired Fat Tree structure. Each ToR switch has four available VLC transceivers to achieve direct wireless communication with other ToR switches. These wireless links organize ToR switches as a random regular graph. Thus the wireless random network and wired Clos network coexist in TIO. To efficiently utilize both wireless and wired links, and to minimize network delay, we present a flow scheduling model to optimize link utilization under a batched traffic pattern. We introduce related definitions and symbols as follows.

Given a data center network $G=(V,E)$, where V and E denote the node set and edge set, respectively. Edge $e \in E$ has capacity $c(e)$. $F=\{f_1, f_2, \ldots, f_\delta\}$ represents δ flows injected into G. For each flow, $f_i=(s_i, d_i, b_i)$, s_i and d_i denote its source node and destination node, respectively, and b_i denotes its traffic demand. The variable $\xi_i(e)$ defines the fraction of flow f_i along link e, where $\xi_i(e) \in [0,1]$ in case the flow can be split among multiple paths, and $\xi_i(e) \in \{0,1\}$ otherwise (i.e., single path routing).

Definition 1. Given G and F, we define the utilization rate of link e as

$$U(e) = \frac{\sum_{i=1}^{\delta} \xi_i(e) \cdot b_i}{c(e)} \tag{1}$$

Note that any $U(e)$ falls into a constant interval [0, 1]. Specifically, if no flow passes through link e, then its utilization rate is 0. The utilization rate is 1 when link e is fully used.

Definition 2. We define the utilization rate of a path P as

$$U(P) = \max U(e), \ \ e \in P \tag{2}$$

The utilization rate reflects the congestion condition. On the basis of $U(P)$, we can locate the bottleneck in a given path and decide whether a path is capable of supporting a given flow.

With the above definitions, we can address the scheduling problem of batched flows and propose the corresponding scheduling algorithm in the following subsections.

4.3 Scheduling the batched flows

We first define the Batched Flow Scheduling (BFS) problem, and formulate it as a linear programming.

Definition 3. Given a network $G(V, E)$ and a batch of flows F, the aim of BFS is to balance traffic load and minimize network congestion. We accordingly formulate the BFS problem as follows:

$$\text{Minimize} \sum_{e \in E} U(e)^2,$$

$$\sum_{i=1}^{\delta} \xi_i(e) \cdot b_i \leqslant c(e), \forall e \in E \tag{3}$$

$$\sum_{e \in \mathrm{out}(s_i)} \xi_i(e) - \sum_{e \in \mathrm{in}(s_i)} \xi_i(e) = 1, \forall i \tag{4}$$

$$\sum_{e \in \mathrm{in}(d_i)} \xi_i(e) - \sum_{e \in \mathrm{out}(d_i)} \xi_i(e) = 1, \ \forall i \tag{5}$$

$$\sum_{e \in \mathrm{in}(\nu)} \xi_i(e) - \sum_{e \in \mathrm{out}(\nu)} \xi_i(e) = 0, \ \nu \neq s_i, d_i, \forall i \tag{6}$$

In the above formulation, i is an integer in the range $[1, \delta]$. Let in(ν) and out(ν) denote the set of edges (ω, ν) that end at node ν and the set of edges (ν, ω) that start at node ν, respectively, where $\omega \in V$, and $(\omega, \nu), (\nu, \omega) \in E$. Equation (3) ensures that the sum of all flows routed over a link does not exceed the links' capacity. Equation (4) indicates that a flow must exit its source node completely, and Eq. (5) indicates that a flow must enter its destination node entirely. Equation (6) ensures that the amount of a flow entering an intermediate node is the same that exists in the node.

In BFS problem, if fractional flows are allowed, then the problem can be solved in polynomial time through linear programming, and we can obtain its accurate optimal solution. However, if flows need to be routed along a single path (the value of $\xi_i(e)$ is either 0 or 1), the BFS problem of producing an integer flow that satisfies all demands (Integer Linear Programming) is NP-hard even for only two flows. Thus, we design a lightweight algorithm to efficiently derive a reasonable solution. For any flow $f_i \in F$, three kinds of paths exist between its source node and destination node: $k^2/4$ wired, one hybrid, and $k^2/4$ wireless paths. We denote the set of these available paths as $\mathcal{P}(f_i)$.

Algorithm 1 shows the insight of the greedy strategy. For each flow f_i, we first calculate its candidate path set $\mathcal{P}(f_i)$ (Line 1), which contains $k^2/4$ wireless links, $k^2/4$ wired links, and one hybrid link. Then, the utilization rate of each candidate path is derived (Line 4). The path in $\mathcal{P}(f_i)$ with a minimal path utilization rate is selected to support f_i (Lines 5 and 6). After accommodating f_i, the capacity of each link in the selected path should decrease by the traffic demand b_i of flow f_i (Line 7). The algorithm executes $O(\delta \times (k^2 + 2))$ time consumption to derive the candidate path sets for all flows in F, and additional $O(\delta \times (k^2/2+1))$ time consumption to select paths for F. Hence, the total computation complexity can be calculated as $O(\delta \times k^2)$.

Algorithm 1 BFS-solution

```
Input: $G$, $F$
Ouput: The solution of BFS problem $S_{\mathrm{batch}}$
1: For each $f_i \in F$, derive $\mathcal{P}(f_i)$;
2: $S_{\mathrm{batch}} = \varnothing, i = 1$;
3: while $i \leqslant \delta$ do
4:     Calculate $U(P)$ for all $P \in \mathcal{P}(f_i)$;
5:     $P_0 = \mathrm{argmin}(U(P))$;
6:     $S_{\mathrm{batch}} = S_{\mathrm{batch}} \cup \{P_0\}$;
7:     $c(e) = c(e) - b_i, e \in P_0$;
8: end while
9: return $S_{\mathrm{batch}}$;
```

5 Performance Evaluation

In this section, we evaluate the performance of our TIO. We first introduce the settings and evaluation methodologies. Then, we compare TIO, Jellyfish, and Fat Tree in terms of topological properties and network performance. Finally, the proposed congestion-aware scheduling method is compared with the widely used Equal-Cost Multi-Path routing (ECMP).

5.1 Setting and methodology of evaluation

We realize the TIO, Jellyfish, and Fat Tree with NS3, a professional network simulator. Given the value of k, the number of racks in TIO and Fat Tree is $k^2/2$. We construct the Fat Tree based on the method in Ref. [22] and generate the TIO as we described in Section 3. The Jellyfish is built with the same number of racks according to the literature[13]. The bandwidth of each VLC links is set as 10 Gbps. For all networks, according to setting in Refs. [23, 24], the link delay is set as 1 ms. With the above settings, we first compare the topological properties of the three DCNs, and the routing complexity of the wireless, wired, and hybrid paths. For these DCNs, we then compare their network performance in terms of throughput and latency.

In the evaluation of network performance, we consider the following two traffic patterns introduced in Refs. [25, 26]:

Random Uniform traffic (RU). In this pattern, source racks select destination racks randomly. Thus, packets are distributed evenly along the whole network.

Random non-uniform traffic (Hot Region, HR). In HR, the first 12.5% of the network (racks with smallest identifiers) bears 25% of the traffic, and the remaining 75% of the traffic is uniformly spread throughout the whole network. This pattern is used to evaluate the network performance under an unbalanced workload.

To prove the effectiveness of our flow scheduling method, we evaluate the network performance of TIO under both ECMP and our scheduling method. Specifically, we measure the throughput and packet loss rate at different network sizes.

5.2 Topological properties

To compare the topological properties of TIO, Jellyfish, and Fat Tree, we measure two metrics, namely, aggregate bandwidth and Average Path Length (APL). Figure 4a shows the aggregate bandwidth of the three topologies, and they are normalized against the bandwidth of TIO. Obviously, the bandwidth is in decreasing order among TIO, Fat Tree, and Jellyfish. As the network size increases with the value of k, the bandwidth gap between Jellyfish and the other two also increases, while the relative gap between TIO and Fat Tree narrows. This result occurs because the aggregate bandwidth is related to the link number. The number of wired links in Fat Tree is $k^3/2$, and the number of wireless links in Jellyfish is k^2. Given that TIO is the combination of Jellyfish and Fat Tree, the number of links is $k^3/2+k^2$, which equals the sum of the links in Jellyfish and Fat Tree.

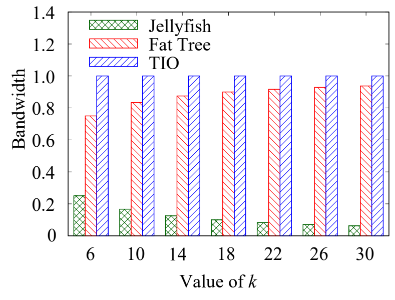
The length of a routing path denotes the number of hops from the source to the destination along the routing path[27]. Using general shortest path algorithms can find the shortest path. Figure 4b illustrates the APL of the three topologies at different network scales. The APL value from large to small is in the order of Jellyfish, Fat Tree, and TIO. For the same number of racks, the APL also has a correlation with link number. Fat Tree has $k/2$ times the link number of Jellyfish. Thus, with the increase in k, the gap between Fat Tree and Jellyfish widens.

We further measure the time consumption of the three routing schemes in TIO, namely, wireless, wired, and hybrid routing. As depicted in Fig. 4c, the time consumption of wireless routing increases from 31 ms to 61 ms, when k ranges from 6 to 30. The time consumption of hybrid routing shows the same changing trend as wireless routing, and it varies from 7 ms to 55 ms. Evidently, wired routing has low time consumption, which is stable and irrelative to the value of k, due to the special hierarchical structure of Fat Tree.

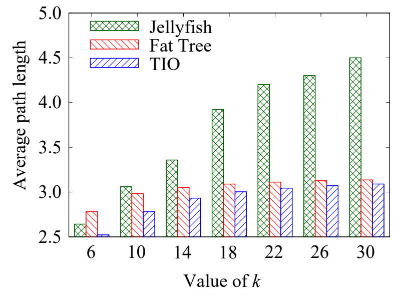
We also compare the connectivity of TIO's pod-level graph under different network sizes. With each k, the generation progress of TIO is conducted multiple times to deduce the configuration strategy of wireless links. The connectivity of the pod-level graph is measured as the number of links, and it is normalized against the corresponding complete graph. The results are illustrated in Fig. 4d, where TIO_1, TIO_2, and TIO_{10} denote the configuration strategy selected after one, two, and ten rounds, respectively. The connectivity increases with the increase in rounds, and decreased with the increase in the network scale.

5.3 Network performance

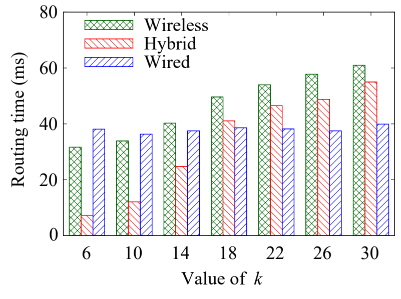
In this section, we evaluate the network performance of TIO, Jellyfish, and Fat Tree in terms of network



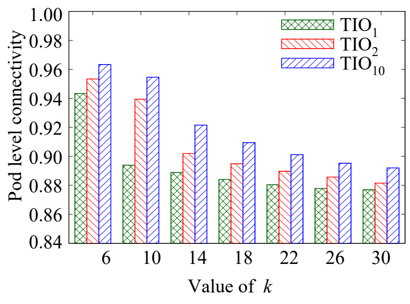
(a) Bandwidth



(b) Average path length



(c) Routing complexity



(d) Pod level connectivity

Fig. 4 Topological properties of TIO.

throughput and latency in the case of the ECMP flow scheduling method. The latency is depicted as the average completion time of all flows. Under each of the two traffic patterns, i.e., RU and HR, we measure the throughput and latency of different network scales. To reveal the impact of flow size on network performance, we use two flows with average sizes of 4 MB and 12 MB, respectively. In each test, the network throughput is normalized against Jellyfish to show their relative size.

5.3.1 Network performance under random uniform traffic

In the setting of RU traffic pattern, we inject k^3 batched random flows to the three topologies to evaluate their network performance. The source and destination servers of each flow are selected randomly. Under the RU pattern, flows disperse evenly in the whole network.

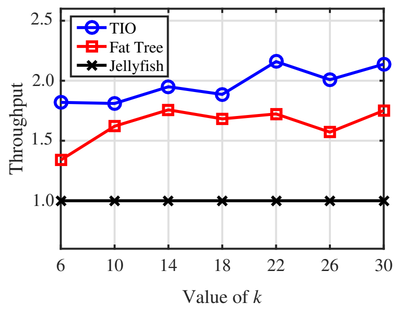
Figures 5a and 5b illustrate the throughput with flow sizes of 4 MB and 12 MB, respectively. The throughputs of different topologies are normalized against Jellyfish. We adjust the network scale by increasing k from 6 to 30. We can see that TIO offers the maximum throughput under both flow sizes, followed by Fat Tree and Jellyfish. With the increase in flow size, the gap between Jellyfish and the other two topologies narrows. When the flow size is fixed as 4 MB, the throughputs of TIO and Fat Tree are 1.96 and 1.63 times greater than that of Jellyfish, respectively, while this ratio decreases to 1.76 and 1.59 under 12 MB flow size, respectively.

Figure 6 shows the average packet delivery latency. The measured packet delivery latency reflects the length of the routing path. Evidently, Jellyfish causes the largest latency among the three topologies, and its latency also grows fastest with the network scale. When the flow size increases from 4 MB to 12 MB, all these topologies consume more time to deliver packets. Given that the upper surface of the racks is not big enough, the number of VLC transceivers that are used to construct the wireless Jellyfish in TIO is fixed at 4 for each rack, not increasing with k. Thus, Jellyfish has the worst performance among the three topologies due to its smallest number of links.

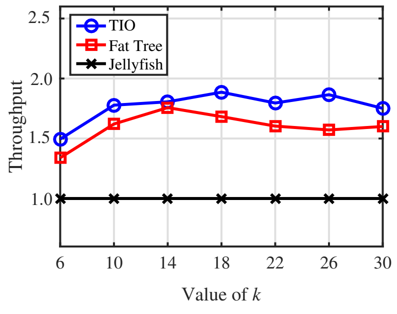
5.3.2 Network performance under hot region traffic

In the setting of the HR traffic pattern, we generate random and non-uniform flows in the three topologies. Thus, we can evaluate the performance of each topology under an unbalanced workload. Specifically, the first 12.5% (racks with lowest identifiers) bears the 25% of the traffic, and the remaining 75% of the traffic is uniformly spread throughout the whole network. In addition, the amount of flows is also k^3.

In this experiment, we set the flow size as 4 MB

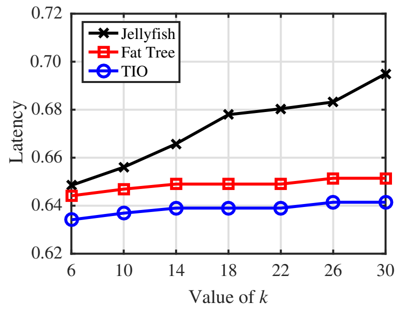


(a) Using 4 MB flows

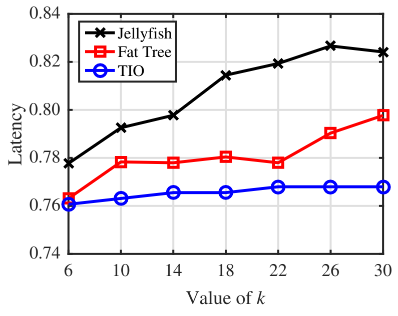


(b) Using 12 MB flows

Fig. 5 Throughput under RU traffic pattern.



(a) Using 4 MB flows



(b) Using 12 MB flows

Fig. 6 Latency under RU traffic pattern

and 12 MB and measure the throughput and latency under diverse network scales by ranging k from 6 to 30. Figures 7 and 8 show the evaluation results. Figure 7 depicts the throughput normalized against Jellyfish, and Fig. 8 shows the average packet delivery latency. Under the HR traffic pattern, the throughput and latency show the same changing trend as the RU traffic pattern. TIO achieves larger throughput and lower latency under both flow sizes. However, the unbalanced workload affects the performance of the three topologies. Compared with the RU traffic pattern, both throughput and latency obtain poor results.

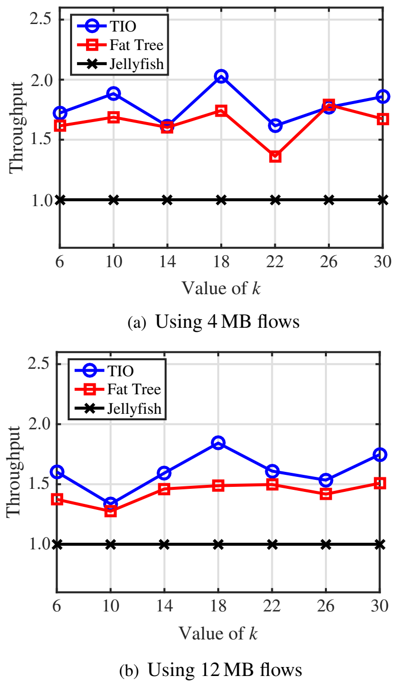


Fig. 7 Throughput under HR traffic pattern.

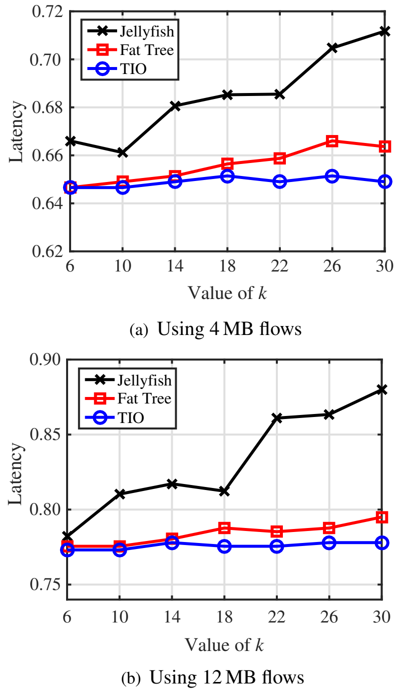


Fig. 8 Latency under HR traffic pattern.

5.4 Impact of congestion-aware flow scheduling

The above evaluations demonstrate the benefits of TIO over the other two topologies with the ECMP flow scheduling method. However, the topological benefits of TIO have not been fully exploited by the existing ECMP methods. Thus, we compare the performance of ECMP with our scheduling method, BFS, under different sizes of TIO.

We injected k^3 batched flows into TIO, where k ranges from 6 to 42. The throughput and packet loss rate are measured with the two scheduling methods. Results are shown in Fig. 9. The throughput is normalized against ECMP. We can see that the ECMP method leads to worse throughput and packet loss rate. By contrast, our scheduling method offers 1.43 times of the throughput of ECMP on average. In addition, the packet loss rate of BFS almost reduces to 0 because $k=18$, while the packet loss rate of ECMP is always high. Obviously, our scheduling method improves the performance considerable because it can provide more candidate paths and distributes the flows widely in TIO.

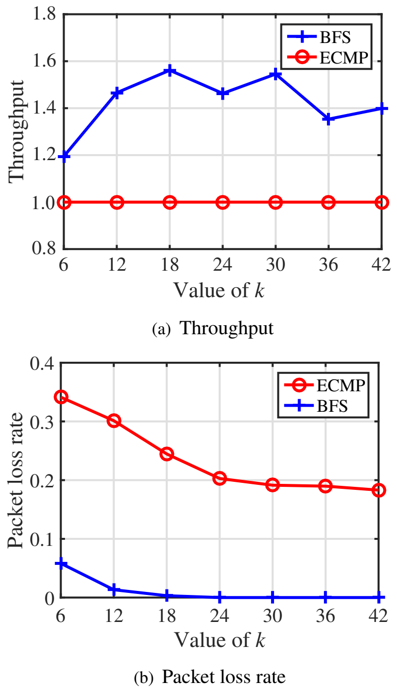


Fig. 9 Performance of scheduling methods.

6 Conclusion

In this paper, we propose TIO, a hybrid DCN that integrates the wireless Jellyfish and wired Fat Tree seamlessly. It employs VLC links to construct a wireless and stable network. Unlike other hybrid designs, TIO is plug-and-play, given that it abandons the complex control mechanism on optical devices. It is also easily deployable. TIO reasonably distributes VLC links in the space above racks; thus, it does not require a mirror or other devices to relay light signals, thereby making it easy to deploy and maintain. To further exploit the benefits of TIO, we propose a hybrid routing scheme and flow scheduling method for batched flows. Evaluations show that TIO outperforms Fat Tree and Jellyfish in both topology properties and network performance, and the flow scheduling method also improves its performance considerably.

Acknowledgment

This work was partially supported by the National Natural Science Foundation for Outstanding Excellent Young Scholars of China (No. 61422214), the National Natural Science Foundation of China (No. 61772544), the National Key Basic Research and Development (973) Program of China (No. 2014CB347800), the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (No. 2016JJ1002), the Guangxi Cooperative Innovation Center of Cloud Computing and Big Data (Nos. YD16507 and YD17X11), and the NUDT Research Plan (No. ZK17-03-50).

References

[1] D. K. Guo, J. J. Xie, X. L. Zhou, X. M. Zhu, W. Wei, and X. S. Luo, Exploiting efficient and scalable shuffle transfers in future data center networks, *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 997–1009, 2015.

[2] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, et al., Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, 2015.

[3] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, Helios: A hybrid electrical/optical switch architecture for modular data centers, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 339–350, 2010.

[4] G. H. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, c-Through: Part-time optics in data centers, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 327–338, 2010.

[5] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, RotorNet: A scalable, low-complexity, optical datacenter network, in *Proc. Conf. of the ACM Special Interest Group on Data Communication*, Los Angeles, CA, USA, 2017, pp. 267–280.

[6] H. H. Bazzaz, M. Tewari, G. H. Wang, G. Porter, T. S. E. Ng, D. G. Andersen, M. Kaminsky, M. A. Kozuch, and A. Vahdat, Switching the optical divide: Fundamental challenges for hybrid electrical/optical datacenter networks, in *Proc. 2^{nd} ACM Symp. on Cloud Computing*, Cascais, Portugal, 2011, p. 30.

[7] K. Ranachandran, R. Kokku, R. Mahindra, and S. Rangarajan, 60 GHz data-center networking: Wireless⇒ Worry less? https://www.researchgate.net/publication/260388834_60_GHz_Data-Center_Networking_Wireless_Worry_ less, 2008.

[8] A. S. Hamza, J. S. Deogun, and D. R. Alexander, Wireless communication in data centers: A survey, *IEEE Commun. Surv. Tutor.*, vol. 18, no. 3, pp. 1572–1595, 2016.

[9] A. S. Hamza, S. Yadav, S. Ketan, J. S. Deogun, and D. R. Alexander, OWCell: Optical wireless cellular data center network architecture, in *Proc. 2007 IEEE Int. Conf. on Communications*, Paris, France, 2017, pp. 1–6.

[10] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, Firefly: A reconfigurable wireless data center fabric using free-space optics, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 319–330, 2014.

[11] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P. A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, ProjecToR: Agile reconfigurable data center interconnect, in *Proc. 2016 ACM SIGCOMM Conf.*, Florianopolis, Brazil, 2016, pp. 216–229.

[12] A. Singla, Designing data center networks for high throughput, PhD dissertation, University of Illinois, Champaign, IL, USA, 2015.

[13] A. Singla, C. Y. Hong, L. Popa, and P. B. Godfrey, Jellyfish: Networking data centers randomly, in *Proc. 9^{th} USENIX Conf. on Networked Systems Design and Implementation*, San Jose, CA, USA, 2012, p. 17.

[14] L. Chen, K. Chen, Z. H. Zhu, M. L. Yu, G. Porter, C. M. Qiao, and S. Zhong, Enabling wide-spread communications on optical fabric with megaswitch, in *Proc. 14^{th} ACM/USENIX Symp. on Networked Systems Design and Implementation*, Boston, MA, USA, 2017, pp. 577–593.

[15] Y. T. Xia, X. S. Sun, S. Dzinamarira, D. M. Wu, X. S. Huang, and T. S. E. Ng, A tale of two topologies: Exploring convertible data center network architectures with flat-tree, in *Proc. Conf. of the ACM Special Interest Group on Data Communication*, Los Angeles, CA, USA, 2017, pp. 295–308.

[16] S. Louvros and D. Fuschelberger, VLC technology for indoor lte planning, in *System-Level Design Methodologies for Telecommunication*, N. Sklavos, M. Hübner, D. Goehringer, and P. Kitsos, eds. Springer, 2014, pp. 21–41.

[17] L. L. Luo, D. K. Guo, J. Wu, T. Qu, T. Chen, and X. S.

Luo, VLCcube: A VLC enabled hybrid network structure for data centers, *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2088–2102, 2017.

[18] A. Roy, H. Y. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, Inside the social network's (datacenter) network, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, 2015.

[19] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, A cost comparison of datacenter network architectures, in *Proc. 6th Int. Conf.*, Philadelphia, PA, USA, 2010, p. 16.

[20] S. Vijay and K. Geetha, A survey on visible light communication appliances used in inter-vehicular and indoor communication, *Int. J. Appl. Eng. Res.*, vol. 11, no. 7, pp. 4893–4897, 2016.

[21] Y. Cui, S. H. Xiao, X. Wang, Z. J. Yang, C. Zhu, X. Y. Li, L. Yang, and N. Ge, Diamond: Nesting the data center network with wireless rings in 3D space, in *Proc. 13th Usenix Conf. on Networked Systems Design and Implementation*, Santa Clara, CA, USA, 2016, pp. 657–669.

[22] M. Al-Fares, A. Loukissas, and A. Vahdat, A scalable, commodity data center network architecture, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[23] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, Dctcp: Efficient packet transport for the commoditized data center, in *Proc. ACM SIGCOMM*, New Delhi, India, 2010.

[24] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. G. Wang, D. Wetherall, and D. Zats, TIMELY: RTT-based congestion control for the datacenter, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 537–550, 2015.

[25] J. M. Camara, M. Moreto, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martinez, and J. Navaridas, Twisted torus topologies for enhanced interconnection networks, *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 12, pp. 1765–1778, 2010.

[26] Y. D. Qin, D. K. Guo, L. L. Luo, G. Y. Cheng, and Z. L. Ding, Design and optimization of VLC based small-world data centers, *Front. Comput. Sci.*, doi: 10.1007/s11704-018-7315-6.

[27] D. K. Guo, J. Wu, Y. H. Liu, H. Jin, H. H. Chen, and T. Chen, Quasi-Kautz digraphs for peer-to-peer networks, *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 1042–1055, 2011.

Yudong Qin received the BS degree from National University of Defense Technology, Changsha, China, in 2016. He is currently working towards the MS degree in College of Systems Engineering, National University of Defense Technology, Changsha, China. His research interests include data centers and software-defined networks.

Deke Guo received the BS degree from Beijing University of Aeronautics and Astronautics in 2001, and the PhD degree from National University of Defense Technology in 2008. He is currently a professor in the College of Systems Engineering at National University of Defense Technology. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Guoming Tang received the bachelor and master degrees from National University of Defense Technology, China, in 2010 and 2012, respectively, and the PhD degree from University of Victoria, Canada, in 2017. He is currently an assistant professor at National University of Defense Technology, China. His research interests include datacenter power management, green computing, and machine learning and optimization.

Bangbang Ren received the BS and MS degrees from National University of Defense Technology (NUDT) in 2015 and 2017, respectively. He is currently a PhD student in NUDT. His research interests include software-defined network, data center network, and network function virtualization.