VLCcube: A VLC Enabled Hybrid Network Structure for Data Centers

Lailong Luo, Deke Guo, Jie Wu, *Fellow, IEEE,* Sujan Rajbhandari, Tao Chen, Xueshan Luo

Abstract—Recent results have made a promising case for offering oversubscribed wired data center networks (DCN) with extreme costs. Inter-rack wireless networks are drawing intensive attention to augment such wired DCNs with a few wireless links. Inspired by the promise of easy deployment and plug-and-play, we present VLCcube, a novel inter-rack wireless solution that extends the design of wireless DCN into three further dimensions: (1) all inter-rack links are wireless; (2) there is no imposition of any infrastructure-level alteration on wired production data centers; and (3) it should be plug-and-play, without any need of additional mechanical or electronic control operations. This vision, if realized, will lead to increased flexibility, reduced reconstructing cost, simplified configuration and usage, and outstanding compatibility with existing wired DCNs. Previous proposals, however, are opposed to the last two design rationales. To achieve this vision, the proposed VLCcube augments Fat-Tree, a representative DCN in production data centers, by organizing all racks into a wireless Torus structure via the emerging visible light links. We further present the topology design, hybrid routing, and flow scheduling schemes for VLCcube. Extensive evaluations indicate that VLCcube outperforms Fat-Tree significantly under the existing ECMP flow scheduling scheme, irrespective of the undergoing traffic pattern. Moreover, the performance of VLCcube can be significantly promoted by our congestion-aware flow scheduling scheme. More precisely, compared to ECMP, our flow scheduling scheme makes VLCcube achieve $\times 1.50$ throughput under batched flows, $\times 2.21$ and $\times 2.59$ throughput under two different kinds of online flows.

Index Terms—data center networks, inter-rack network, visible light communication, throughput, packet loss rate.

1 Introduction

Data centers have emerged as infrastructures for online applications and infrastructural services. Thousands of servers and switches are interconnected via a specific data center network (DCN). DCNs can be roughly divided into two categories. The first category is wired DCNs, each of which connects all switches and servers with wired links via cables, fibers or twisted-pair links. Fat-Tree [1] and VL2 [2] fall into this category. The second one is wireless DCNs, which employ wireless links to argument a wired DCN or organize servers and switches as a fully wireless network structure [3] [4] [5].

Wired DCNs suffer from inherent challenges. First, they are either overprovisioned with good performance but high cost, or oversubscribed with low cost but poor performance. Second, it is extremely costly and complicated when expanding a wired data center. Third, they cause vast cabling and maintenance cost [6]. Fourth, large-scale wired DCNs usually adopt multiple-level structures. As a result, two servers, which across racks, must employ the upper-level links to communicate with each other, even if they are very close physically.

To eliminate the non-trivial cost and increase the flexibility during the expanding process of any wired production DCN, several wireless DCNs are proposed at the inter-rack level. As depicted in Fig.1, the racks are connected with the introduced wireless links. Typically, the radio frequency (60GHz) [4] and free-space-optical (FSO) communication techniques [3] are employed to establish an inter-rack wireless network. These proposals can considerably improve the performance of any existing wired DCNs in terms of bandwidth and packet latency [7]. Moreover, the wireless links can be dynamically reconfigured to meet the demand of the undergoing flows.

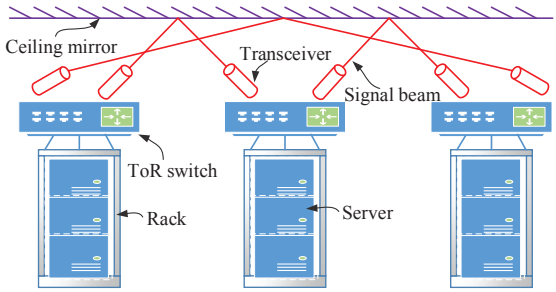


Fig. 1. An example of wireless DCN, in which the racks are interconnected with wireless links.

Inspired by the promise of easy-deployable and plug-and-play, we envision a radically different design of inter-rack wireless network, which should simultaneously concern the following three design rationales: (1) all inter-rack links are wireless; (2) without imposing any infrastructure-level alteration on the existing wired production data centers; and (3) the inter-rack wireless network is plug-and-play and has no need of additional mechanical or electronic control operations.

- *Lailong Luo, Deke Guo, Xueshan Luo and Tao Chen are with the Science and Technology Laboratory on Information Systems Engineering, National University of Defense Technology, Changsha Hunan 410073, China.*
- *Jie Wu is with the Department of Computer and Information Science, College of Science and Technology, Temple University, Philadelphia, Pennsylvania, USA.*
- *Sujan Rajbhandari is with the Department of Engineering Science, University of Oxford, Oxford, United Kingdom.*

This vision, if realized, will lead to unprecedented benefits for wireless DCNs. Firstly, it ensures high flexibility and low cabling cost of the network by introducing wireless links on demand. Secondly, it simplifies the configuration and usage process of the inter-rack wireless network, due to bringing no additional mechanical or electronic control operations. Such simplification makes the inter-rack wireless network extremely compatible with existing wired DCNs. Thirdly, it alleviates the burden of managing and maintaining a data center. Once those wireless links are established, they will work permanently without additional control operations.

Existing proposals on the inter-rack wireless network focus on the flexible reconfiguration of links. Such proposals, however, do not consider other two essential design rationales. First of all, they have to update or even reconstruct the deployment environment of existing production data centers. For example, prior proposals using 60GHz, as well as FSO communication, have to decorate the ceiling to be a huge mirror to achieve over-the-horizon communication [3] [4]. Besides, to realize flexible reconfiguration, dedicated optical devices are required, e.g., ceiling mirrors, plano/bi convex lens [3]. Moreover, they impose frequent and complicated control on wireless devices and peripheral equipment when configuring a wireless link.

In this paper, we propose VLCcube to achieve the above three design rationales simultaneously. VLCcube augments Fat-Tree, a representative production wired DCN, by organizing all racks into a wireless Torus structure via the emerging visible light communication (VLC) techniques. Hence, it is a hybrid network structure of data center by seamlessly integrating the wired Fat-Tree and wireless Torus together. Although the 60GHz and FSO communication techniques are also suitable for VLCcube in theory, we prefer the VLC since it is becoming a promising choice for the next-generation wireless technology by offering low cost, unregulated bandwidth and ubiquitous infrastructures support. Inherently, VLC links eliminate the peripheral devices except VLC transceivers and need no additional mechanical or electronic control. The contributions of this paper are summarized as follows:

- We design VLCcube, a hybrid DCN structure, which employs VLC wireless links to interconnect all racks in a Fat-Tree data center as a wireless Torus. The topology construction strategy is well designed to ensure high connectivity and low average path length. As an easy-deployable and plug-and-play hybrid DCN, VLCcube realizes the three design rationales at the same time.
- To fully exploiting the topological properties, we present a hybrid routing scheme for VLCcube to jointly utilize both wired and wireless links. To further improve the network performance, we design a light-weight method to address the optimized flow scheduling problem. The method can efficiently derive an outstanding solution for batched as well as online flows.
- Comprehensive experiments are conducted to measure the performance of VLCcube. The results indicate that VLCcube outperforms Fat-Tree significantly under the existing ECMP flow scheduling scheme, irrespective of the used traffic pattern. Compared to ECMP, our congestion-aware flow scheduling scheme make VLCcube achieve better performance, i.e., $\times 1.50$ throughput under batched flows, $\times 2.21$ and $\times 2.59$ throughput under two kinds of online flow patterns.

The remainder of this paper is organized as follows. Section 2 summarizes prior designs of data centers. Section 3 proposes the VLC enabled hybrid network structure, VLCcube, for data centers. Section 4 puts forward the hybrid routing method for VLCcube, and designs the congestion-aware flow scheduling method. We evaluate the performance of VLCcube in Section 5 and discuss the potential limitations and solutions in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

Due to the essential status of DCNs, a huge body of work has been conducted to improve network performance. The representative DCNs can be classified into two categories, i.e., the wired DCNs and the wireless DCNs.

2.1 Wired DCNs

Typically, the wired DCNs take advantage of the merit of excellent topologies, e.g., Torus, Hypercube, Kauzt, Small-world, etc. We further assort the existing wired topologies into four fine-grained classes, i.e., switch-centric data centers, server-centric data centers, modular data centers, and random data centers.

In switch-centric data centers, routing and interconnection are realized by switches, which form dedicated structured topology, such as generalized hypercube, Torus, compound graph, tree and so on. Fat-Tree [1], F10 [8], VL2 [2] belong to this category. With the development of optical communication, optical packaging technology is introduced into switch-centric DCNs [9]. These optical links improve bandwidth greatly, but the associated control strategy becomes complex.

Note that switches and routers are expensive, while commodity server and mini-switch are cheap; hence, it is cost-saving to build a DCN just with servers and mini-switches. In server-centric data centers, routing and interconnection are realized by servers since servers are competent to cache and forward flows. Usually, server-centric DCNs are recursively defined and extended level by level. BCube [10], DCell [11] are all server-centric DCNs, but their network order are limited by the count of NIC ports at each server.

To ease the development of data centers, module has replaced racks as the basic building block of large-scale data centers. These modules integrate the power system, cool system and thousand servers inside a container. By further interconnecting a given number of such modules via a dedicated topology, an efficient, controllable, and elastic data center can be built. MDCube [12] and uFix [13] are two representative proposals. They utilize the remaining NICs at servers to interconnect those modules systematically.

For random DCNs, random links interconnect remote nodes together, hence, they shorten the network diameter

[14]. Typically, Jellyfish [15] and Scafida [16] are proposed based on the random regular graph and scale-free network, respectively. The advantage of random DCN is the characteristic of incremental expansion, which means that we can add servers one by one other than level by level. Routing in such random topologies, however, is difficult and time-consuming.

2.2 Wireless DCNs

Recently, based on the developing wireless communication techniques, such as 60GHz communication, laser communication, free-space-communication, etc, wireless technologies are investigated for DCNs. Therefore, the cabling cost will be considerably eliminated and the network bandwidth will be increased.

In literature [17], a remote wireless channel between any pair of racks can be established by reflecting wireless signals via a mirror from source to destination. FireFly [3] goes further, it forecasts the traffic demand and adjusts the topology dynamically in a short time period. Wireless DCNs supports unicast transmission well, but fails to accomplish other transmission models such as broadcast, multicast and shuffle. But it is true that, as an complementary interconnection method, the wireless links speed up the network significantly.

Besides, the feasibility of building fully wireless DCNs is verified [7]. Deployed with two transceivers, each server can communicate with others independently. Then, the modified servers are stacked and interconnected as cylindrical racks. By networking the racks as a specific topology, a wireless DCN is established. This proposal also calls for accurate direction control of the transceivers. Besides, since only two transceivers are deployed at each server, the expansion of network order will lead to drastically increase of network diameter.

Undoubtedly, the totally wireless DCNs are costly to deploy and loses the merit of wired topologies, e.g., regularity, easy-routing and stability. Thus, aiming to integrate the superiority of both wireless network and wired network, we employ the VLC links to connect the existing wired DCN to be a hybrid one.

3 The Design of VLCcube

We first discuss the feasibility and interference issue of interconnecting racks using VLC links, and accordingly design a novel VLCcube topology. It seamlessly augments the wired data center Fat-Tree, using a wireless inter-rack Torus network.

3.1 Feasibility of introducing VLC links into DCNs

For VLC, transmitting data is achieved by intensity modulation of visible spectrum lighting emitting diodes (LEDs) or laser diodes (LDs). On-Off keying modulation scheme, where "ones" and "zeros" are represented by the presence or absence of light, is the simplest form of digital communication [18] [19]. To employ the VLC links, three vital issues have to be concerned, including the data rate, transmission distance, and the accessibility of devices.

Data rate. By employing those high switching frequency LEDs, a single color VLC link can realize considerably high data rate up to 3 Gbps [20]. Such devices could potentially deliver data rates in the order of 10 Gbps by using RGB triplet. Besides, a single laser beam can even achieve 9 Gbps data rate by employing the 450-nm GaN LDs [21]. Hence, we believe that the data rate of VLC links is capable of transmission in data centers.

Transmission distance. The LED based VLC links can achieve about 10 Gbps data rate within 10 meters, which is sufficient to interconnect two close racks inside a data center. We notice that a project named Rojia prolong the distance of VLC to 1.4 kilometers, but with limited data rate [22]. Besides, the LD based VLC links can realize fast long-distance communication (in the order of kilometers) with high data rate [23], due to its outstanding directionality. Hence, the LED based VLC links can be employed as the short links, while the LD based VLC links are competent to the long distance transmission in DCNs.

Accessibility. The off-the-shelf full-duplex VLC devices, i.e., transceivers, are developed and released [22] [24]. A development platform called MOMO [25] has delivered API and SDK for users to customize their VLC-based applications. For example, the VLC techniques have seamlessly integrated into a platform of internet of things. Moreover, pureLiFi [24] provides the opportunity for customers to rapidly develop and test VLC applications for cost-effective, high-speed data communication solutions by using commercial LED infrastructures.

Accordingly, it is reasonable to employ the VLC links to augment wired DCNs, without incurring additional cabling cost or modifying the hosting environment of data centers.

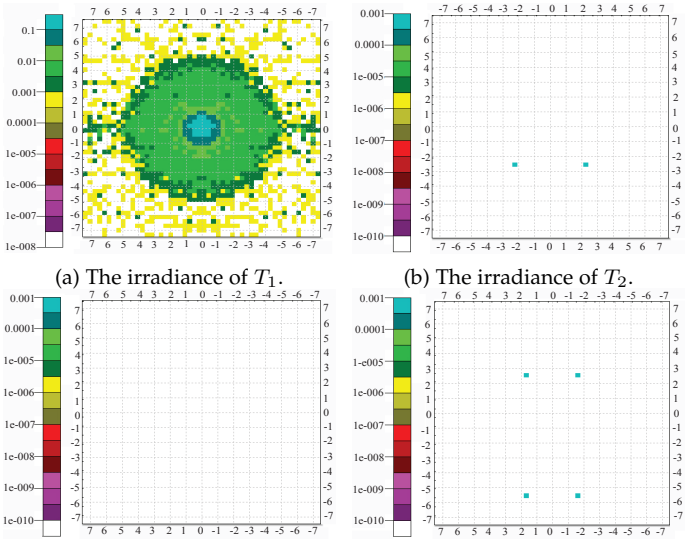
3.2 The interference among transceivers

The benefits of VLC links motivate us to employ them to augment the wired inter-rack networking in data centers. However, the interference is an essential obstacle when utilizing VLC links.

Typically, on the top of each rack, a few VLC transceivers should be setup such that the ToR switches can be organized as a dedicated wireless topology. Given a rack R, when multiple neighboring racks send data to it simultaneously, interference occurs if multiple transceivers on R can perceive the light from different source racks but fail to distinguish them.

To evaluate the VLC interference, we conduct simulations using a professional optical software, i.e., TracePro70 [26]. As depicted in Fig.3, we place four receivers with orthogonal orientations on the top of a rack, which are denoted as T_1, T_2, T_3 and T_4, respectively. Then, a batch of visible light is emitted towards T_1 from three meters away. The irradiance map of each receiver can identify how much light has been detected by other receivers. If T_2, T_3 and T_4 detect intensive light, it demonstrates that the interferences to them are prominent.

Fig.2(a) depicts the result observed by T_1. It is obvious that the receiver detects the majority of the emitted light, and the central part of the receiver captures the most of them. Due to the scattering, some light deviates from the central line; hence, the non-central areas can also detect



(a) The irradiance of T_1. (b) The irradiance of T_2.

(c) The irradiance of T_3. (d) The irradiance of T_4.

Fig. 2. The irradiance of each receiver in the simulation.

the light. By contrast, as shown in Fig.2(b), Fig.2(c) and Fig.2(d), the other three receivers can hardly capture the light since the normalized irradiance is only 0.001 in several points in such figures. We also note that, Fig.2(c) records the least irradiance at T_3. Consider that T_3 is right behind T_1 and it is difficult for the light to pass by T_1 to reach T_3. Consequently, the light towards T_1 results in limited interference to other three receivers. This observation shows that deploying four transceivers on the top of a rack is feasible and will bring negligible interference. Accordingly, we will design the wireless topology of VLCcube, where each rack owns just four VLC transceivers.

3.3 Topology design of VLCcube

Inside a data center, each server connects to the ToR switch inside a rack. All racks usually form a hierarchical network structure by using additional upper level wired links and network devices, rather than connect with each other directly using wired links. For this reason, we aim to interconnect all ToR switches according to a dedicated wireless network structure. In this paper, we adopt the widely used Fat-Tree as an example of wired DCN and augment it with wireless Torus network structure. In this way, we achieve a hybrid VLCcube, which can seamlessly integrate both wired and wireless DCNs.

As depicted in Fig.3, all racks in wired Fat-Tree DCN are further interconnected via VLC links to form a two-Dimensional wireless Torus, with m racks in each row and n racks in each column. On the top of each rack, four VLC transceivers are deployed towards four orthogonal directions, such that the interference can be restricted at the lowest level. Note that, the wired part of VLCcube is a Fat-Tree topology, and we just connect the ToR switches via VLC links. Let k denote the number of ports of each switch, which is usually even. Thus, VLCcube accommodates k pods, each of which has $k/2$ ToR switches and $k/2$ aggregation switches. Thus, $k^2/2$ ToR switches are involved in the wireless part of VLCcube.

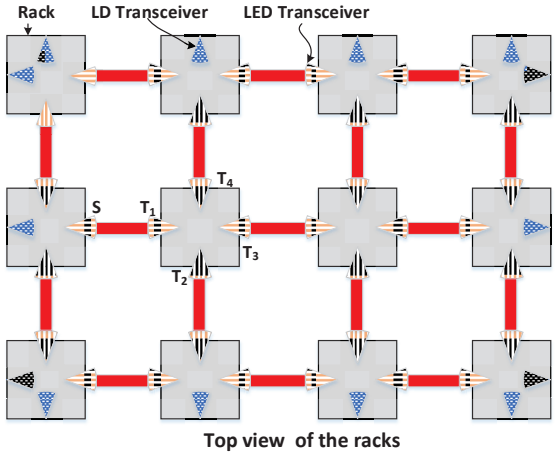


Fig. 3. The inter-rack wireless network of VLCcube. All racks in Fat-Tree are interconnected as a wireless Torus via VLC links.

Note that, in Fig.3, all of racks in each dimension should be interconnected as a loop. Thus, two racks at the ends of each row (column) should be connected directly. To achieve such long-distance connections, the LD-based VLC links are employed. By contrast, the short-distance connection between any pair of adjacent racks is enabled by the LED-based VLC links. In Fig.3, we only depict the short links, while the long links are omitted for the easy of presentation. Note that the LED-based VLC links are not competent to support long-distance connections, since the signal strength degrades sharply during the diffusion process.

Another important issue is how to avoid shadowing in VLCcube. Consider that there exists no barrier between any pair of adjacent racks. Those LED transceivers used to offer short wireless links will not suffer from the shadowing problem. On the contrary, multiple racks lie between the end racks, which are connected by long links. As shown in Fig.3, one or two LD transceivers are deployed on the top of those involved racks. We classify the LD transceivers into two categories, i.e., the horizontal LD transceivers connecting the racks to form a loop in each row, and the vertical LD transceivers connecting the racks to form a loop in each column. Without loss of generality, we spatially isolate the LED transceivers, the horizontal LD transceivers, and the vertical LD transceivers, e.g., 0.1m, 0.3m and 0.5m, respectively, such that they are deployed in diverse planes. Through such carefully considerations, the interference and shadowing problems of VLC links can be well tackled.

In fact, we can design the hybrid VLCcube topologies in two ways. A straightforward way is to augment the wired network structures with a wireless 2D Torus directly. By contrast, a more advisable method can further promote the topology by jointly consider the wireless 2D Torus and the wired Fat-Tree.

3.3.1 *Independent topology design of wireless Torus*

We notice that, the wireless 2D Torus can be attached to the existing wired Fat-Tree directly. Without loss of generality, we assume that, in a k-pod Fat-Tree, there are k rows and $k/2$ columns, i.e., m=k, n=$k/2$. In each dimension, every rack enables wireless links with its neighboring racks. As

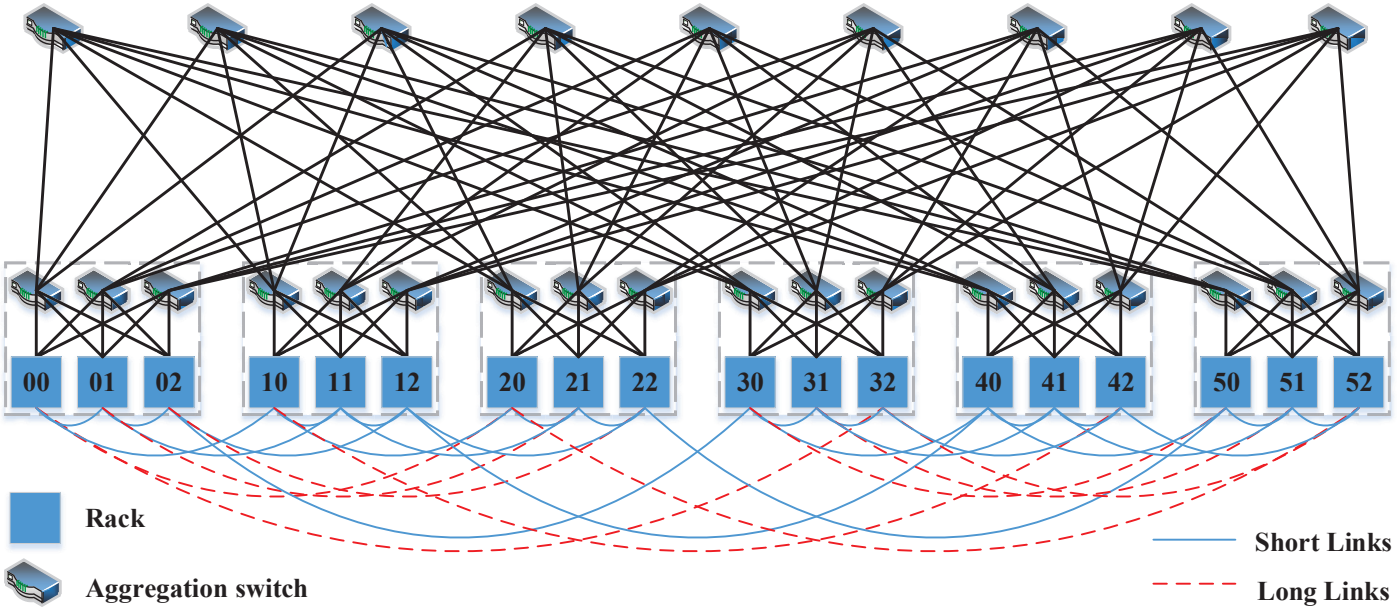


Fig. 4. An expressive example of VLCcube when k=6. The two dimensional coordinate xy denotes the y-th rack in the x-th pod, $x \in [0, k-1]$ and $y \in [0, k/2-1]$. Note that, the short and long links are launched by LED and LD transceivers, respectively.

a result, the diameter of the 2D Torus is $0.75k$, which is proportional to the number of k.

Moreover, in the resulted wireless 2D Torus, given a rack in the i-th pod, as shown in Fig. 5, each rack has four neighbors. However, these neighbors come from at most three different pods. Thus, at the pod level, the i-th pod is connected to three other pods, but fails to communicate with other pods directly. Fig.5 depicts a toy example when k=6. The wireless 2D Torus is constructed independently without concern of the placement of racks. As a result, a 6×3 2D Torus is generated. Note that, the diameter of the resulted Torus is 4. However, the pod level logic graph is not a completed graph; hence the path from a rack in pod 0 to pod 4 must employ a relay rack in pod 1 or pod 3.

The resultant topology of VLCcube with this design methodology is shown in Fig.4. On one hand, the ToRs and switches are interconnected as a Fat-Tree by the wired links. On the other hand, the racks are also interconnected as a wireless Torus by the introduced VLC links. To realize these VLC links, four transceivers are established on the top of each rack. Note that, the short-distance communication is achieved by the LED-based VLC links, and the long-distance communication is realized by the LD-based VLC links. It is true that the $k^2/2$ wireless Torus improves the connectivity of racks and increases the variety of paths. However, the improvement of performance can be further enhanced if the arrangement of racks can be optimized.

3.3.2 Joint topology design of wireless Torus and wired Fat-Tree

To fully utilize the benefits of VLC wireless links, we optimize the topology of VLCcube by integrating the 2D Torus with Fat-Tree seamlessly. To reach this goal, two important issues must be well tackled, including the settings of m and n, and the placement of racks in VLCcube.

Parameter setting. Note that all nodes in each dimension of a 2D Torus form a loop structure; hence, the network

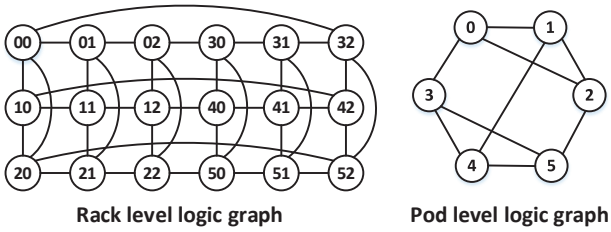


Fig. 5. The logic graph of VLCcube without considering placement of racks.

diameter by $(m+n)/2$. For this reason, VLCcube aims to minimize the network diameter of the used Torus structure by inferring reasonable configurations of m and n. Additionally, the total number of remote VLC links in VLCcube is $m+n$, and such a few long links are more difficult to establish, compared to those short VLC links. This issue further motivates VLCcube to minimize $m+n$ for eliminating unnecessary remote VLC links. If a 2D Torus is designed to accommodate $k^2/2$ racks, the parameters m and n should be bounded by the inequation $m \times (n-1) < k^2/2 \leq m \times n$.

Theorem 1. In VLCcube, the optimal setting of m is calculated as $\lceil \sqrt{k^2/2} \rceil$. The value of n depends on $k^2/2$. If $(m-1)^2 < k^2/2 \leq m \times (m-1)$, n is $m-1$; in contrast, when $m \times (m-1) < k^2/2 \leq m^2$, n is set the same as m, i.e., $\lceil \sqrt{k^2/2} \rceil$.

Proof: The best settings of m and n should minimize the value of $m+n$. Note that we have $m+n \geq 2 \times \sqrt{m \times n} \geq 2 \times \sqrt{k^2/2}$. Thus, $m+n$ reaches its minimum value only when $m=n$. Considering the inequation $m \times (n-1) < k^2/2 \leq m \times n$, we derive the relationship between m, n and k. □

Placement of racks. As for the placement problem, we note that, during the design stage, the placement of racks and cables can be jointly optimized with the respect of wireless Torus network structure. Indeed, the path length between any pair of ToRs is either two or four hops in Fat-

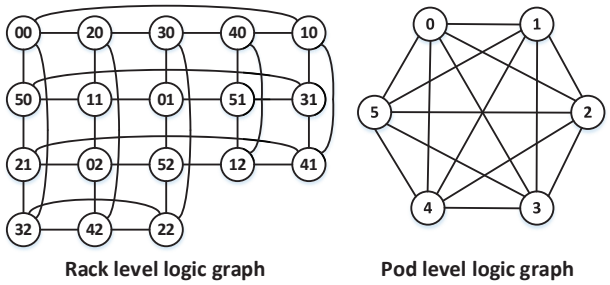


Fig. 6. The logic graph of VLCcube in the rack level and pod level.

Tree. Hence, VLCcube aims to shorten those four hops wired communication as just one hop wireless communication by reconsidering the location of racks. Given m and n, we further concern the best placement of racks for supporting the inter-rack wireless Torus network. Note that, in Fat-Tree, if two racks fall into the same pod, the path length between them is 2; otherwise, four hops are required. VLCcube targets at shortening the four hops of wired path as one hop wireless path. That is, all VLC links are utilized to connect those racks across pods, rather than those racks inside a same pod.

To ease the presentation of the placement strategy, we first introduce the identifier for each rack, which consists of two parts. The prefix, ranking from 0 to k, denotes which pod this rack belongs to. The suffix, ranking from 0 to $k/2$, identifies the rack in each pod. For example, the identifier 51 refers to the second rack in the sixth pod.

We further define the pod level logic graph, which regards a pod in VLCcube as a node. If there exist one or multiple VLC links between a pair of pods, an edge is added between them in the logic graph. Fig.6 depicts an example of the wireless part of VLCcube, with k=6, m=5 and n=4. Accordingly, the pod level logic graph is derived. Typically, we measure the connectivity of the pod level logic graph by counting the number of links in the graph. In Fig.6, 15 links interconnect 6 pods, thus the connectivity of the pod level graph is 15. Given the value of k for VLCcube, the connectivity of its pod level logic graph is no more than $k\times(k-1)/2$.

With the above definitions and given k, m and n, we design three steps to construct the 2D wireless Torus, which may be an incomplete one, as shown in Fig.6.

- Step 1, allocating the prefixes. For each prefix $x\in[0,k]$, we randomly allocate it to $k/2$ ToR switches in the ToR level logic graph since each pod contains at most $k/2$ ToR switches. The only constraint is that any rack cannot hold the same prefix as its four neighbors. If conflicts occur, repeat this step until all prefixes have been mapped into the graph.
- Step 2, calculating the suffixes. In the ToR level logic graph, a suffix is introduced to differentiate those racks in a same pod. Note that the suffix of each rack ranges from 0 to $(k-1)/2$.
- Step 3, improving the connectivity of the pod level logic graph. We repeat the above two steps multiple rounds, and then pick the solution that leads to the highest connectivity of the pod level logic graph.

However, as stated before, the Fat-Tree is actually installed as a, w.l.o.g, $k^2/2$ array. Obviously, we need to transform the existing $k^2/2$ array to be a $m\times n$ one. Typically, two steps are needed. First of all, $(k-m)\times k/2$ racks must be moved such that the racks are placed as a $m\times n$ Torus physically. Then we deploy the placement strategy of racks logically by rewiring the cables between the aggregation switches and the racks. Undoubtedly, these adjustments suffer from dedicated time-consumption and labour cost. But we believe these once-and-for-all augments are worthwhile to improve the network performance.

Validity of the generation steps. We further prove that our generation method can result in a correct VLCcube structure.

Theorem 2. When $k\geq4$, the above generation method can successfully generate a VLCcube such that each pod appears $k/2$ times in the ToR level logic graph.

Proof: In step 1, we allocate k pods randomly under the constraint that each link connects different pods. If each pod is associated with one color, the proof of Theorem 2 is equivalent to prove that k colors can color the graph successfully. In fact, the ToR level graph of VLCcube is a 4-regular graph, whose chromatic number is 4, which means 4 kinds of colors are enough to color the graph. That is, when $k\geq4$, we can always find out a legal placement strategy. Thus, Theorem 2 is proved. □

Note that, aforementioned generation steps must ensure that the pod level logic graph is connected. Otherwise, VLCcube does not work well since those VLC links cannot reach every pod.

Theorem 3. The pod level logic graph resulting from the above steps are connected.

Proof: Note that the ToR level logic graph is an incomplete 2D Torus, which is a connected graph. That is, a rack identified as xy can find a path to its destination rack uv. If we map this path to the pod level logic graph, it is just the path from pod x to pod u. Thus, Theorem 3 is proved. □

Theorems 2 and 3 ensure the rationality of the generation steps. Step 3 further ensures the connectivity of the pod level logic graph by selecting the best one after executing the first two steps multiple rounds. The behind insight is that by conducting the processes more rounds, we are more possible to achieve the better solution [27]. We will evaluate the performance of such a generation method in Section 5.3.

From the view of topology design, VLCcube integrates the topological characteristics of both Fat-Tree and 2D Torus, e.g., scalability, constant degree, multi-path, and fault-tolerance. Moreover, VLCcube is easy-deployable and plug-and-play, since only four transceivers are needed to deploy for each rack, and no further control operations are required during the usage process after the deployment. More importantly, VLCcube achieves the inter-rack wireless network, without any modification to the hosting environment of a Fat-Tree data center.

4 Routing and Congestion Aware Flow Scheduling in VLCcube

For any pair of ToR switches, wired paths, wireless paths and hybrid paths coexist in VLCcube. The routing algorithms for wired paths and wireless paths can be found in

literatures [1] and [14]. We focus on designing the hybrid routing between racks. To minimize the network congestion, we define a congestion-aware flow scheduling model and design scheduling algorithms for the batched and online traffic patterns.

4.1 Hybrid routing scheme in VLCcube

In VLCcube, the wireless Torus and the wired Fat-Tree are tightly integrated. Thus given a pair of racks, we can search out a path with both wireless and wired links. Specifically, given the source rack xy and destination rack uv, we first deduce the path in the pod level logic graph from pod x to pod u. We then embody each link on the pod level path by choosing a reasonable wireless link from the ToR level logic graph. At last, the involved wired links will be added to the path. This straightforward method, however, incurs high time-complexity and is impractical. The reason is that, since the pod level logical graph is not structured, the Dijkstra algorithm will be employed to calculate the shortest path at the cost of $O(k^2)$ computation-complexity.

As the increase of k, the resultant time-consumption will not be acceptable for DCNs. Note that the introducing of the hybrid path in VLCcube is to shorten the length of wired path, such that the transmission will be accelerated. From this point of view, some hybrid paths resulting from the above routing algorithm may not realize such a design goal. Hence, we prefer to only searching out the hybrid path, which can shorten the wired path between a pair of racks. In this way, unnecessary computation will be avoided. The maximum length of wired paths in VLCcube is 4, and the VLC links connect all pods directly to shorten these paths of 4 hops. Given a flow, only if its source or destination rack is one end of a VLC link between the source pod and the destination pod, the deduced hybrid path for this flow will be 3 hops.

Bearing this insight in mind, given a pair of racks xy and uv, our hybrid routing scheme consists of the following two steps. Firstly, we judge whether xy (uv) launches a VLC link towards the pod u (x). If not, the routing scheme will be stopped; otherwise, move to next step. Secondly, without loss of generality, we assume that a VLC link connects xy with uw directly, and then derive the aggregation switch, which must be added into the hybrid path to relay uw to uv. Note that the racks and the aggregation switches in a pod form a complete bipartite graph. Hence, the aggregation switch can be selected randomly, and a reasonable hybrid path will be achieved.

For example, the hybrid path from rack 00 to rack 11 consists of only 3 hops in Fig.4, since pod 1 and pod 0 are connected by a VLC link from rack 11 to rack 01. Therefore, the 4-hop wired path between 00 and 11 will be shortened as a 3-hop hybrid path, including one aggregation switch but no core switch. Then, the routing scheme selects an aggregation switch from pod 0 to relay the flow from rack 00 to rack 01. The generated hybrid path includes two wired links in pod 0 and a VLC link from rack 01 to rack 11.

Note that, there are $k/2$ racks in each pod, and the routing scheme needs to check whether rack xy (uv) directly connects with one rack in pod u (x) via a VLC link. Thus the time complexity of the first step is $O(k)$. Additionally, the second step consumes constant time. Therefore, the time complexity of this hybrid routing scheme is $O(k)$.

4.2 Problem formulation of flow scheduling

We introduce the VLC links to augment the existing DCNs in VLCcube. The insight is to organize ToR switches as a wireless incomplete Torus via VLC links. To utilize both wired and wireless links efficiently and minimize the delay in the network, we present a flow scheduling model to optimize the link congestion rate, under both batched and online traffic patterns. Consider that there are four available transceivers on each rack. Thus, any rack can communicate with its four neighboring racks simultaneously. We first introduce related definitions and symbols as follows.

Let $G{=}(V, E)$ denote a data center network, where V and E are the node set and link set, respectively. Additionally, $F{=}\{f_1, f_2, \cdots, f_\delta\}$ denotes δ flows injected into G. For each flow $f_i{=}(s_i, d_i, b_i)$, s_i, d_i, and b_i denote the source node, destination node and traffic demand, respectively. Typically, ϕ records a scheduling strategy, which is responsible to derive the routing path for each flow in F.

Definition 1. Given F and ϕ, we define the congestion rate of an arbitrary link e as:

$$C_F^\phi(e){=}t(e)/c(e), \tag{1}$$

where $t(e)$ denotes the amount of traffic passing through link e, and $c(e)$ records the capacity of link e. Note that any $C_F^\phi(e)$ falls into a constant interval $[0, 1]$. Specifically, if none of flows passes through link e, its congestion rate is 0. The congestion rate is 1 when link e is fully used.

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

$$C_F^\phi(P){=}\max C_F^\phi(e),\ where\ e \in P. \tag{2}$$

Accordingly, based on $C_F^\phi(P)$, we can locate the bottleneck in a given path and decide whether a path is capable to serve a given flow.

Based on the above fundamental definitions, we consider the scheduling of both batched flows and online flows, and then propose the corresponding scheduling algorithms in the following subsections, respectively.

4.3 Scheduling the batched flows

To handle the scheduling of batched flows properly, in this section, we first formulate the problem and define the congestion coefficient that will be employed later. Then, a greedy algorithm is introduced and the correctness of the algorithm is given.

4.3.1 Formulation and definitions

Definition 3. (SBF: scheduling batched flows) Given $G(V, E)$ and a set of flow transmissions F, the goal of batched flow scheduling is to find a reasonable flow scheduling strategy ϕ^* such that $Z{=}\max\limits_{e\in E} C_F^{\phi^*}(e)$ is minimized.

We accordingly formulate the SBF problem as follows:

Minimize Z

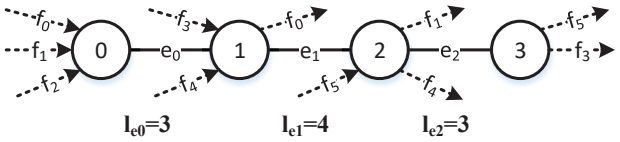


Fig. 7. An example of congestion efficient of link and path.

$$\sum_{f:f\in out(s_i)} b_f = b_i + \sum_{f:f\in in(s_i)} b_f \qquad \forall i \tag{3}$$

$$\sum_{f:f\in in(d_i)} b_f = b_i + \sum_{f:f\in out(d_i)} b_f \qquad \forall i \tag{4}$$

$$\sum_{f:f\in in(x)} b_f = \sum_{f:f\in out(x)} b_f \quad \forall i, \forall x \notin \{s_i, d_i\} \tag{5}$$

In the above formulation, i is an integer in the range $[0, \delta]$. Let $out(v)$ and $in(v)$ denote the set of the outgoing and incoming flows at node v in VLCcube, respectively. Eq.3, Eq.4 and Eq.5 ensure that each flow just transmits along one path. The SBF problem is an Integer Linear Programming (ILP) problem, which is a well-known NP-hard problem. It cannot be solved in polynomial time. A naive method is to search all the potential solutions, and then select the solution with the minimum Z. However, this naive method is time-consuming as well as inefficient. Thus, we design a lightweight algorithm to derive a reasonable solution. For any $f_i \in F$, we find out the three kinds of routing paths in VLCcube and denote them as $\mathcal{P}(f_i)$. Actually, $\mathcal{P}(f_i)$ contains $k^2/4$ wired paths, one hybrid path and one wireless path. To derive the flow scheduling strategy for F, we design a greedy heuristic algorithm based on the concept of congestion coefficient.

Definition 4. Given a set of flows F, each flow $f_i \in F$ has a set of candidate routing paths $\mathcal{P}(f_i)$. The congestion coefficient of a link $e \in E$, denoted as l_e, is the total amount of candidate paths passing through it under all flows in F.

Definition 5. For any routing path $P \in \mathcal{P}(f_i)$ of any flow f_i in F. The congestion coefficient of P, denoted as l_P, can be calculated as $l_P = \sum l_e$, where $e \in P$.

Fig.7 presents the calculation insight of congestion coefficients. Note that, 4 switches (labeled as 0, 1, 2, 3, respectively), 3 links (including e_0, e_1, and e_2), and 6 flows (denoted as f_i, $i \in [0, 5]$) are involved in the figure. Link e_0 is employed by a candidate path of three flows (f_0, f_1 and f_2); hence, the congestion coefficient of link e_0 is 3. Note that link e_1, is involved in the candidate path of 4 flows (f_1, f_2, f_3 and f_4). Accordingly, the congestion coefficient of link e_1 is derived as 4. Similarly, a candidate path of f_2, f_3 and f_5 contains link e_2; hence the congestion coefficient of link e_2 is 3. Note that the candidate path of f_2 is ended at switch 3, and the flow will not be forwarded to other switches in the figure. Additionally, the maximum congestion coefficient of links in the path from switch 0 to switch 3 is 4, thus the congestion coefficient of this path is 4.

Indeed, the congestion coefficient of link e or path P indicates the probability that multiple flows employ it, respectively. Hence, l_P is an index for our greedy algorithm to decide whether flow f_i should select path P. To be specific, we should select the path with the least congestion coefficient among all paths in $\mathcal{P}(f_i)$.

Algorithm 1 SBF-solution (S_{batch})

Require: Input the model of SBF problem.
1: Initialize S_{batch} as empty;
2: For each $f_i \in F$, derive $P(f_i)$;
3: Calculate the congestion coefficient of each link in VLCcube;
4: **for** $i < \delta$ **do**
5: Calculate the congestion coefficient of each path in $\mathcal{P}(f_i)$;
6: Select the path with the least congestion coefficient;
7: Add the chosen path into S_{batch};
8: Label the links on the chosen path as used;
9: **return** The solution of SBF problem S_{batch};

4.3.2 Algorithm and proof

Based on the congestion coefficient, Algorithm 1 shows the insight of the greedy strategy. For each flow, we first calculate its $k^2/4+2$ candidate routing paths (Line 2). Then, the congestion coefficient of each link in VLCcube is derived (Line 3). For each flow f_i, we calculate the congestion coefficient for each of its candidate routing paths and choose the path with the least congestion coefficient to serve that flow (Line 4-8). The algorithm takes $O(\delta \times (k^2+k+4))$ time-consumption to derive the candidate routing paths for all flows in F, and additional $O(\delta \times (k^2/4+2))$ time-consumption to decide which routing path should be utilized by each flow. Hence, the total computation complexity can be calculated as $O(\delta \times k^2)$.

The congestion coefficient of a link e means there are up to l_e flows may employ the link. The congestion coefficient of a path P demonstrates that at most l_P flows may pass through at least one link along the path. If no scheduling strategy is utilized, any path $P \in \mathcal{P}(f_i)$ has the equal probability to be chosen to transmit f_i. Note that our algorithm selects the path with the least congestion coefficient to transmit a dedicated flow. If the probability of congestion for a selected path is proportional to its congestion coefficient, the correctness of employing the congestion coefficient as an index in our algorithm can be proved. Unfortunately, calculating the exact probability of congestion for a path is rather complicated. Hence, as an alternative, we calculate the probability that a path or a link transmits more than two flows, since congestion may occur only if more than 2 flows share a common link or path.

Theorem 4. In VLCcube, given a flow $f_i \in F$, e is an arbitrary link in the network, the probability that e is utilized by f_i is:

$$p_e^{f_i} = \begin{cases} = & 0, \quad f_i \notin F_e \\ = & l_e^{f_i}/(k^2/4+2), \quad f_i \in F_e \end{cases} \tag{6}$$

where F_e records the set of flows that may employ the link e, $l_e^{f_i}$ denotes the congestion coefficient of the link e caused by f_i , since more than one candidate routing paths of f_i may cover that link e.

Proof: Note that, if no scheduling strategy is utilized, any path $P \in \mathcal{P}(f_i)$ has the equal probability to be chosen

to transmit f_i. For flow f_i, if number of $l_e^{f_i}$ paths in $\mathcal{P}(f_i)$ pass through link e, we have $p_e^{f_i}$=$l_e^{f_i}/(k^2/4+2)$. Otherwise, flow f_i never utilizes that link, and the probability is 0. Thus, Theorem 4 is proved. □

Theorem 5. In VLCcube, for any flow $f_i \in F$, η counts the number of flows that pass through a link e, then we have:

$$p_e^F(\eta{=}0) = \prod_{f_i \in F} (1-p_e^{f_i}) \tag{7}$$

$$p_e^F(\eta{=}1) = \sum_{f_i \in F} [p_e^{f_i} \times \prod_{f_j \in F-f_i} (1-p_e^{f_j})] \tag{8}$$

$$p_e^F(\eta \geq 2) = 1-p(\eta{=}0)-p(\eta{=}1) \tag{9}$$

Proof: Given a flow set F, such flows are independent for whether employ a link e or not. Hence, $p(\eta{=}0)$ and $p(\eta{=}1)$ can be calculated easily. Thus, Theorem 5 is proved. □

Theorem 6. Consider a flow $f_i \in F$, let η counts the number of flows that pass through a path $P \in \mathcal{P}(f_i)$, and $E(P)$ denotes the set of links along the path P. For any P, we have:

$$p_P^F(\eta{=}0) = \prod_{e_i \in E(P)} p_{e_i}^F(\eta{=}0) \tag{10}$$

$$p_P^F(\eta{=}1) = \frac{4}{k^2+8} \prod_{e_i \in E(P)} p_e^{F-f_i}(\eta{=}0) + \sum_{e_s \in E(P)} [p_{e_s}^{F-f_i}(\eta{=}1) \times \prod_{e_j \in E(P)-e_s} p_{e_j}^{F-f_i}(\eta{=}0)] \tag{11}$$

$$p_P^F(\eta \geq 2) = 1-p_P^F(\eta{=}0)-p_P^F(\eta{=}1) \tag{12}$$

Proof: For a path $P \in \mathcal{P}(f_i)$, $\eta{=}0$ means none of flows passes any link in path P. While, $\eta{=}1$ is resulted from two situations, i.e., only f_i occupies the path P, or one link in path P has been utilized by another flow $f_j \in F-f_i$. Thus, $p_P^F(\eta{=}0)$ and $p_P^F(\eta{=}1)$ can be calculated. □

According to Theorem 4, Theorems 5 and 6 calculate the probability that none or one flow passes link e and path P. Note that, if $\eta \geq 2$, link e or path P may result in congestion. This will happen when the completion time of the former flow blocks the transmission of the latter flow. Theorems 5 and 6 demonstrate that larger l_e leads to more opportunities that more than 2 flows go through link e or path P, and may cause congestion. Thus the probability that a path P is blocked is proportional to its congestion coefficient l_P. In this way, the correctness of employing the congestion coefficient of a path as an index for Algorithm 1 is certified. Since our greedy algorithm selects the paths with the least congestion coefficient, the congestion rate in VLCcube will be decreased significantly.

4.4 Scheduling online flows

As discussed in [28], flows are not always batched in data centers. In fact, flows are usually uncertain and dynamic. Typically, ϕ^0 depicts an existing flow scheduling strategy, F_N denotes the new arriving flows, and F_O contains the flows that call for retransmission. Accordingly, we update the set of flows as F_1=F_N+F_O. With F_1 as input, we define the online flow scheduling problem as follows:

Algorithm 2 SOF-solution (S_{online})

Require: Input the model of SOF problem.
1: Initialize S_{online} as empty;
2: Calculate the updated routing requests F_1;
3: Update the state of network links and devices;
4: **for** $i < \delta_1$ **do**
5: Search the three kinds of paths from s_i to d_i;
6: Calculate the congestion rate of each path;
7: Select the path with the least congestion rate, i.e., $path_i$;
8: Add $path_i$ into S_{online};
9: **return** The solution of SOF problem S_{online};

Definition 6. (SOF: scheduling online flows) The SOF problem is to deduce a new scheduling strategy ϕ_1 such that the increased link congestion rate is minimized. Let ΔZ=Z_1−Z_0, where Z_1=$\max_{e \in E} C_{F_1}^{\phi_1}(e)$ and Z_0=$\max_{e \in E} C_{F-F_O}^{\phi^0}(e)$, the goal of SOF is to minimize ΔZ.

The SOF problem will be triggered when new flows appear or some existing flows are required to be retransmitted. Note that the SOF problem still subjects to an ILP model, which is similar to the SBF problem. We omit the detailed presentation of the SOF model due to the page limitation.

The SOF problem targets at minimizing Z_1. Thus, it seems that the same strategy, depicted in Algorithm 1, can be utilized to solve the SOF problem. Algorithm 1, however, will be employed frequently due to the dynamic flows, and hence causes unnecessary computation cost. Instead, we only take flows in F_1 into consideration, and propose a greedy flow scheduling strategy for the SOF problem. For each flow in F_1, the insight of our greedy strategy is to employ the path that causes the least link congestion rate.

As depicted in Algorithm 2, the greedy strategy discovers those flows that call for path assignment (Line 2). Typically, it distinguishes the finished flows, the new flows and the failed flows. The algorithm has to know which available links and devices the updated flows can employ. We update the state of the whole network by eliminating finished flows (in Line 3). For each flow, we search all of its wireless path, hybrid path and wired paths according to those algorithms in Section 4.

After deriving the possible candidate routing paths for a flow f_i, we calculate the congestion rate of each path according to the values of b_i and c_i. We then pick the path with the least congestion rate as the final routing path for f_i and add it into S_{online} (Line 4-8). When each flow in F_1 has been assigned a reasonable path, Algorithm 2 returns the result. Algorithm 2 just considers those flows that need to be allocated a routing path. Thus, the algorithm will be executed δ_1 rounds, while the time consumption is $O(k+k^2)$ in each round, due to derive the candidate paths. Thus, the computation complexity of Algorithm 2 is $O(\delta_1 \times k^2)$.

Theorem 7. Algorithm 2 outperforms the traditional ECMP flow scheduling strategy for the online traffic pattern.

Proof: For any flow $f_i \in F_1$, if ECMP is employed, the expectation of congestion rate for f_i is

$$\frac{4}{k^2+8} \sum_{P_j \in \mathcal{P}(f_i)} C_{F_1}^{\phi^*}(P_j) \tag{13}$$

By contrast, the congestion rate of Algorithm 2 for f_i is

$$min\{C_{F_1}^{\phi^*}(P_j)\} \;\; s.t.\; P_j \in \mathcal{P}(f_i) \tag{14}$$

Undoubtedly, for an arbitrary flow f_i, the congestion rate under Algorithm 2 is no more than that under ECMP. Thus, Theorem 7 is proved. □

5 Performance Evaluation

We start with the qualitative comparison between VLCcube with other proposals. As for the qualitative evaluations, we first introduce the settings and methodologies. Then, we compare VLCcube with Fat-Tree, in terms of the topological properties and network performance. Finally, the proposed congestion aware scheduling methods are evaluated against the widely used ECMP. The reported result is the average value over 100 rounds of evaluations for each metric.

5.1 Qualitative comparison

Before discussing the quantitative evaluations, we first compare VLCcube with several wireless data center topologies qualitatively.

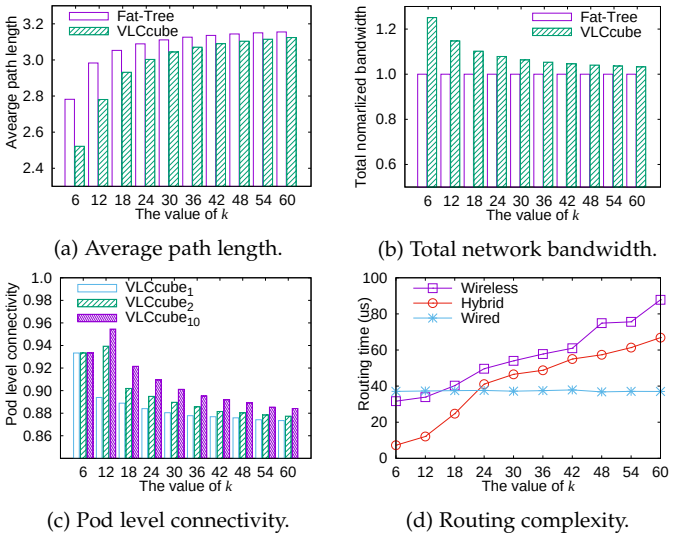
Table 1 depicts the results. Firstly, FireFly, 3D Beamforming (3D BF) and VLCcube employ the Laser, 60 GHz and VLC wireless links to establish another wireless topology over existing racks, respectively. Secondly, the wireless topologies of both FireFly and 3D BF are flexible since the wireless links can be adjusted on demand, according to the traffic pattern. The wireless topology of VLCcube is always fixed, so as to maintain a stable wireless and hybrid topologies over racks. This would avoid the non-trivial cost due to adjusting wireless links and ease the designing of routing schemes. Thirdly, both FireFly and 3D BF will introduce complex infrastructure-level alterations, e.g., redecoration of the ceilings, establishment of various communication components.

Fourthly, to realize the designed wireless topology, complicated mechanical or electronic control operations are required to establish a wireless link in FireFly and 3D BF. On the contrary, VLCcube is plug-and-play and does not suffer from such extra control operations, when establishing each VLC link. Moreover, Firefly and 3D BF must accurately predict the traffic pattern before designing a suitable wireless topology. VLCcube, however, does not need such kind of traffic prediction. It is time-consuming and costly to accurately predict the traffic pattern, especially for those burst and hot-spot traffics.

Therefore, VLCcube offers stable wireless topology, easy routing mechanism at the cost of slightly releasing the requirement of topology flexibility. Additionally, VLCcube meets the three design rationales proposed in this paper and needs not to accurately predict the traffic pattern.

TABLE 1
Qualitative comparison.

Structure	Wireless	Flexible	Alteration	P-a-p	Predict
Firefly	Yes	Yes	Yes	No	Yes
3D BF	Yes	Yes	Yes	No	Yes
VLCcube	Yes	No	No	Yes	No



(a) Average path length. (b) Total network bandwidth.

(c) Pod level connectivity. (d) Routing complexity.

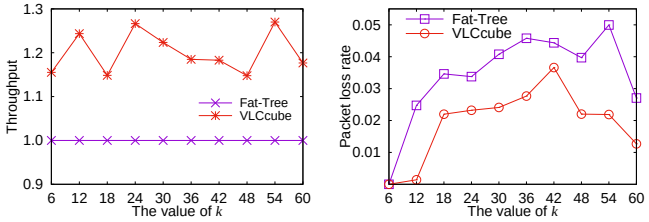
Fig. 8. Topological properties of VLCcube.

5.2 Setting and methodology of evaluations

We realize the proposed VLCcube and Fat-Tree with Network Simulator (NS3). Given the number of k, we generate Fat-Tree according to the rules introduced in [1]. As for VLCcube, we generate it with the steps in Section 3.3. To reveal the essential impact of introducing VLC links to DCNs, we assume that the adjustment of racks is permitted. The bandwidth of each wired link and VLC link is set as 10 Gbps. For both networks, according to the setting in [29] [30], the link delay is set as 1 ms. With the above basic setting, we first compare their topological characteristics, and then evaluate the complexity of routing algorithms for wired paths, wireless paths and hybrid paths. Moreover, we compare their network performance.

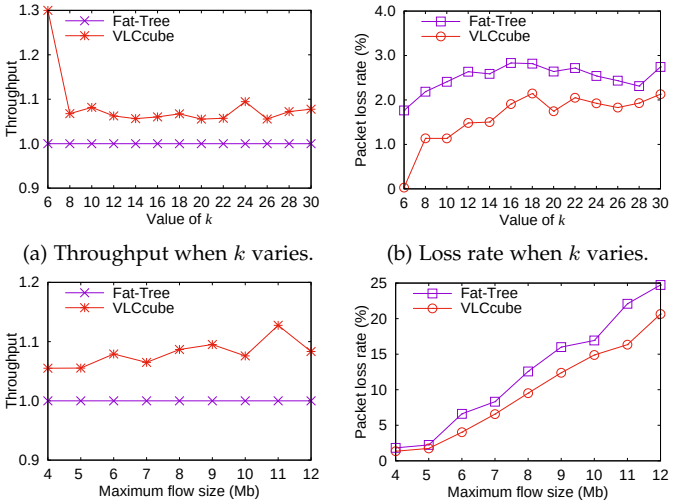
In our evaluations, we consider three traffic patterns: i) Trace flows: the flows are generated by a real data-set from Yahoo!'s data centers [31]; ii) Stride-i flows: a server with id x sends flow to the destination with id $(x+i)\ mod\ N$, where N is the total number of servers; and iii) Random flows: the source and destination of each flow are chosen randomly. The network throughput and packet loss rate, are used to measure the performance of DCNs under diverse traffic patterns.

To prove the effectiveness of our flow scheduling methods, we first evaluate the network performance of VLCcube and Fat-Tree, which both utilize the prior ECMP flow scheduling method. We further measure the performance of VLCcube under our flow scheduling algorithm and the ECMP, respectively. Note that the arrival time of dynamic flows follows a Poisson distribution in the case of online flow scheduling.



(a) Throughput under trace flows. (b) Loss rate under trace flows.

Fig. 9. Network performance under trace flows.



(a) Throughput when k varies. (b) Loss rate when k varies.

(c) Throughput when flow size varies.(d) Lose rate when flow size varies.

Fig. 10. Network performance under stride flows.

5.3 Topological properties of VLCcube

Two topological properties, the average path length (APL) and the total network bandwidth, are measured for VLCcube and Fat-Tree. As shown in Fig.8(a) and Fig.8(b), VLCcube has shorter APL and offers higher network bandwidth than Fat-Tree, due to those VLC wireless links. Note that, the impact of VLC wireless links on the APL exhibits an obvious marginal effect. That is, the APL would be significantly decreased by VLC wireless links for small-scale networks. In fact, given k, there are k^2 VLC links in VLCcube, and the number of both wired and wireless links is $k^3/2 + k^2$. As the increase of k, VLC links contribute less portion to the total number of links. Hence, the impact of wireless links becomes weak.

We conduct the generation progress multiple rounds to deduce the placement strategy for VLCcube. Then the connectivity of the pod level logic graph is normalized and measured as the number of links in a complete graph. In Fig.8(c), VLCcube_1, VLCcube_2 and VLCcube_{10} denote the measured results, under the picked placement strategy with the highest pod level connectivity after one round, two rounds, and ten rounds generations. It is clear that the connectivity reduces along the increase of k, irrespective of the rounds of generation. Additionally, VLCcube_{10} always outperforms other two cases, since the generation algorithm may derive a better solution from more candidates with high probability.

We further measure the consumed time, due to calculate three kinds of routing paths in VLCcube. As depicted in Fig. 8(d) the time-consumption of searching the wireless path increases from 37 microseconds to 87 microseconds, when k grows from 6 to 60. The time consumption of hybrid path routing shows the similar increasing trend and varies from 8 microseconds to 67 microseconds since the time-complexity is $O(k)$. It is clear that the wired path routing causes the least time consumption, which remains in a low level, i.e., 39 microseconds, irrespective of the setting of k. In summary, the time complexity of wired path routing is constant, while that of other two routing algorithms is proportional to the value of k. The time-consumption of the wireless and hybrid paths is only tens of microseconds even k=60, and hence is acceptable in data centers.

5.4 Network performance

In this section, we evaluate the network performance of VLCcube and Fat-Tree in terms of network throughput and packet loss rate in the case of ECMP flow scheduling method. Under each of the three traffic patterns, we vary the network scale by adjusting the value of k, and observe the changing trends of the network throughput and packet loss rate. To reveal the impact of flow size on the network performance, the average size of each flow ranges from 5 Mb to 12 Mb. Note that the size of each flow under the trace-based traffic pattern is always set according to the trace. In each test, the network throughput is depicted as the ratio of the real throughput of between VLCcube and Fat-Tree in each kind of parameter setting.

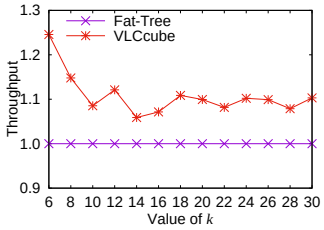
5.4.1 *Network performance under trace flows*

The Yahoo!'s trace records the basic information for each flow in its 6 distributed data centers, including the IP addresses of both source and destination servers, the flow size, the utilized interfaces, etc. We separate those inner data center flows from those flows across data centers by identifying the utilized interfaces [31]. We inject k^3 randomly chosen flows into the VLCcube and Fat-Tree networks to evaluate their performance.

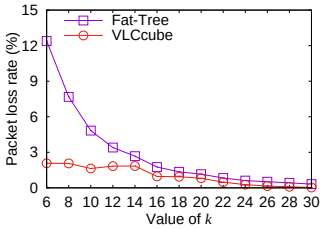
Fig.9(a) and Fig.9(b) plot the performance of VLCcube and Fat-Tree in terms of both throughput and packet loss rate when k varies from 6 to 60. It is clear that VLCcube dominates Fat-Tree by offering more throughput (19.97% more) and causing much less packet loss rate (39.00% less). The reason is that, those VLC links provide more candidate paths for each flow.

5.4.2 *Network performance under stride-k flows*

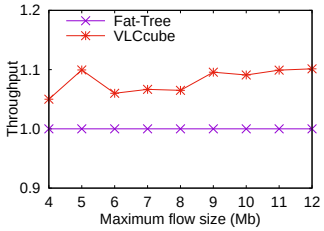
In this experiment, firstly, we set the maximum flow size as 5 Mb. We measure the throughput and packet loss rate under diverse network orders by increasing k from 6 to 30. Fig.10(a) and Fig.10(b) report the evaluation results, when $2\times k^2$ random flows are injected in the networks. With the increase of k, both Fat-Tree and VLCcube are capable of accommodating more flows, thus their throughputs increase rapidly. But VLCcube achieves 8.54% more throughput than Fat-Tree on average. Additionally, as shown in Fig.10(b), the packet loss rate of VLCcube is much less than that of Fat-Tree. We also note that, with the increase of k, both Fat-Tree and VLCcube result in more packet loss rate. The reason



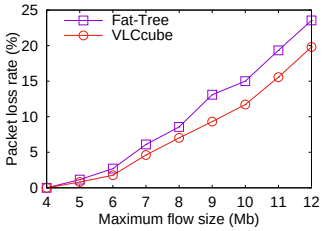
(a) Throughput when k varies.



(b) Loss rate when k varies.

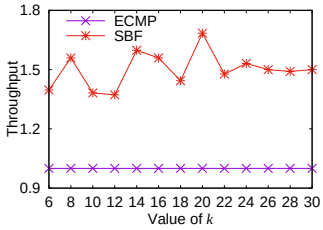


(c) Throughput when flow size varies.

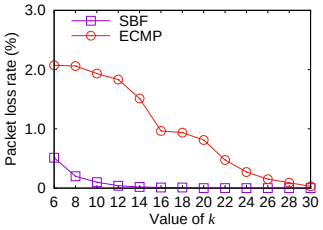


(d) Loss rate when flow size varies.

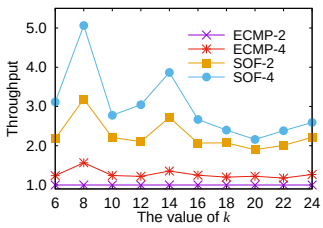
Fig. 11. Network performance under random flows.



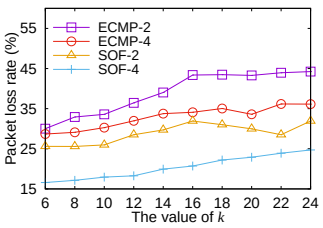
(a) Throughput of batched scheduling.



(b) Loss rate of batched scheduling.



(c) Throughput of online scheduling.



(d) Loss rate of online scheduling.

Fig. 12. The performance of congestion aware scheduling.

is that, the number of injected flows ($2\times k^2$) increases faster than the number of candidate paths for each flow ($k^2/4+2$). As a result, more flows may be dropped.

To measure the impact of flow size, we vary the average size of those $2\times k^2$ flows from 4 Mb to 12 Mb while k=20. We can infer from Fig.10(c) and Fig.10(d) that VLCcube still considerably outperforms Fat-Tree. More precisely, VLCcube increases the network throughput up to 14.31% than Fat-tree even when the maximum flow size is 11 Mb. Reasonably, when the flow size grows, the packet loss rate increases dramatically.

5.4.3 *Network performance under random flows*

In the setting of random traffic pattern, the source and destination server of each flow are all selected randomly. We also introduce $2\times k^2$ flows into the two data center networks.

First of all, we fix the maximum flow size as 5 Mb, and vary the network scale by ranging k from 6 to 30. As shown in Fig.11(a), for both VLCcube and Fat-Tree, the network throughput increases dramatically. VLCcube still outperforms Fat-Tree with 10.80% more network throughput on average. As depicted in Fig.11(b), Fat-Tree always experiences a high packet drop rate, while VLCcube incurs much less packet drop rate. To be specific, VLCcube and Fat-Tree drop 1.01% and 2.92% packets on average, respectively.

We further evaluate the impact of maximum flow size on the network performance when k=20. As shown in Fig.11(c) and Fig.11(d), VLCcube and Fat-Tree must transmit more packet once the flows are scheduled; hence, the throughput increase reasonably as larger flows are injected in the networks. At the same time, the packet loss rate also increases along with the increase of average flow size. We derive from such figures that our VLCcube works better than existing Fat-Tree significantly.

In summary, VLCcube achieves better network performance than Fat-Tree under the three kinds of traffic patterns, when both of them employ the ECMP to schedule flows.

5.5 Impact of congestion aware flow scheduling

Although the above evaluations demonstrate the benefits of VLCcube than Fat-Tree, the topological benefits have not been fully exploited by employing the existing ECMP flow scheduling method. Thus, we compare our congestion aware scheduling method with ECMP under different sizes of VLCcube.

We inject k^3 batched random flows into VLCcube, where k varies from 6 to 24. The network throughput is normalized as the ratio of the network throughput under the ECMP method to that under our SBF method. As depicted in Fig.12(a) and Fig.12(b), ECMP offers less network throughput and causes the worse packet loss rate. By contrast, our SBF method contributes $\times 1.50$ throughput and causes much less loss rate (decreasing to 0 when k=12) than ECMP. The root cause of low loss rate is that our SBF method offers more candidate paths, and disperses the flows as widely as the VLCcube can.

Additionally, the VLCcube schedules online flows using our SOF method. In this case, we vary k from 6 to 24 and schedule k^3 random flows under each configuration of VLCcube. The arrival time of dynamic flows follows a Poisson distribution, whose parameter λ can be adjusted.

Fig.12(c) and Fig.12(d) plot the evaluation results. Note that ECMP-x and SOF-x refer the results under the ECMP and our SOF methods, when λ=x. Our SOF method leads to $\times 2.21$ and $\times 2.59$ throughput than ECMP, while causes only $\times 0.746$ and $\times 0.619$ packet loss, when λ= 2 and λ=4, respectively. Note that, both ECMP and SOF methods in the case of λ=4 outperform that in the case of λ=2. The reason is, less flows will simultaneously arrive in a given time interval as the increase of λ; hence, such flows cause less packet loss.

Consequently, our SBF and SOF flow scheduling methods can improve the performance of VLCcube and realize less congestion rate than the widely used ECMP method.

6 Discussion

In this paper, we augment the existing wired Fat-Tree DCN by introducing the inter-rack wireless network using VLC

links. To fully understand the designing methodology of VLCcube, we discuss the following important issues.

Why VLC links? Plenty of endeavors have been conducted towards the designing of wireless data centers. In this paper, we make the first step to study the feasibility of introducing VLC into data centers. That is, we establish another inter-rack wireless network for a data center with any wired DCN. Such VLC links bring extra advantages. Firstly, compared with the overcrowded RF spectrum, the visible light spectrum occupies hundreds of terahertz of license free bandwidth, which are remains untapped. Secondly, VLC can well reuse the existing lighting infrastructures to realize high-speed communication. That is, VLC is cost-effective as well as energy-saving.

The building methods of VLCcube. In this paper, we point out that VLCcube can be constructed in two different methods. Indeed, the racks in an existing Fat-Tree data center have been placed as a fixed array, e.g., $k^2/2$ array. Hence, a simple method is to remain the arrangement of racks and interconnect all racks as a wireless Torus. However, the resultant hybrid DCN brings limited improvement of network performance. To fully exploit the benefits of the introduced VLC links, we prefer to upgrade all racks from the $k^2/2$ array to a $m \times n$ array, which brings extra adjustment of racks and links. Consequently, the resultant VLCcube can replace more four-hop wired paths with one-hop wireless paths. Additionally, the connectivity among pods is enhanced by those VLC links. In summary, there is a trade-off between the extra cost and the promotion of performance for the second method.

The scalability of VLCcube. It is true that the Fat-Tree DCN lacks high scalability. The network order of Fat-Tree is decided by the value of k. If there are already $k^2/2$ racks in Fat-Tree, any added rack will result in the increase of k. As a result, the scalability of our VLCcube is limited by the employed Fat-Tree strucutre. However, it is not necessary that the design of VLCcube depends only on the Fat-Tree like structures. In the future, VLCcube would consider other scalable wired DCNs, e.g., Jellyfish [15], FBFLY [32] and HyperX [33]. The networking structure of Jellyfish is a random regular graph at the level of rack, while that of FBFLY and HyperX is the generalized hypercube at the level of rack. There are no aggregative or core switches in these DCNs. When VLC links are introduced into such wired DCNs, the resultant hybrid topologies can be expanded easily. On the other hand, we envision the completely wireless DCNs based on VLC links as our future work, which can be expanded on demand. In summary, we make a first step towards designing a hybrid topology by introducing VLC links into wired DCNs, for example Fat-Tree. The methodology proposed in this paper can be used to design other scalable hybrid DCNs if we utilize other scalable wired DCNs instead of the Fat-Tree.

The complexity of routing methods. As a hybrid topology, VLCcube offers wired paths, wireless paths and hybrid paths for any pair of racks. The wired path as well as the wireless path can be simply calculated according to the building rules of Fat-tree and Torus topologies. To speed up the hybrid routing algorithm, we only derive the 3-hop hybrid paths in VLCcube. The time-complexity of the wired routing scheme is constant for the Fat-tree DCN. However, the time-complexity of the wireless and hybrid routing algorithms is $O(k)$. As the evaluation results indicate, the time-consumption for calculating the wireless path and the hybrid path for any pair of racks is within tens of microseconds, which is acceptable for current data center applications. Moreover, once all routing paths for any pair of racks have been derived and kept in involved routing tables, such routing algorithms will seldom be triggered. That is, this kind of latency only happens during the initialization or updating process of the network.

Rethinking of flow model and scheduling algorithms. In VLCcube, due to the existence of multiple paths, it is important to select a proper path for each flow since those paths result in diverse completion time. From the global view, we need to derive a suitable path for every flow, such that the congestion rate is minimized. After allocating given path for each flow, more accurate and fine-grained control mechanisms can be realized by several existing proposals, e.g., Hedera [34], pFabric [35], L2DCT [36], etc. Such transport control mechanisms target at optimizing the flow completion time, using dedicated rules, such as the shortest remaining processing time first, the deadline first, and the smallest flow first, etc.

The evaluation methodology. Our evaluations concentrate on measuring the impact of introducing VLC wireless links into the wired Fat-Tree networking structure for data centers. For fairness, we compare VLCcube with Fat-Tree when both of them employ the ECMP strategy. Then we evaluate the impact of the proposed scheduling algorithms for VLCcube. The comprehensive evaluations do verify the improvement of our VLCcube over Fat-Tree, in terms of both topological properties and the network performance. Besides the evolutions via comprehensive simulations, we also consider the possibility of small-scale deployment of VLCcube system. However, existing VLC products are still incapable of gigabit-level data rate since the high-speed VLC techniques (at the scale of Gbps) are still at the stage of test in the laboratory. As a result, it is inappropriate to compare VLCcube enabled by existing VLC products with Firefly or Fat-Tree currently. We will keep tracking the products of the next generation high-speed VLC links and deploy a prototype of VLCcube system to truly improve the existing wired data centers.

Future work. The future work is mainly of two folds. On one hand, we will consider the design and evaluation of other hybrid DCNs using VLC links, in the settings of other wired DCNs (e.g., BCube, MDcube, Jellyfish, etc). New challenges will occur when integrating the VLC wireless links with other existing wired topologies. On the other hand, we envision a complete wireless data center based on VLC links only.

7 Conclusion

In this paper, we present VLCcube, an easy-deployable, and high-performance hybrid DCN architecture. VLCcube introduces the emerging VLC technique to interconnect all racks together with a wireless Torus network, so as to augment the wired Fat-Tree network. The introduced VLC links can decrease the APL and enhance the network bandwidth. To exploit the benefits of VLCcube, we further

design dedicated flow scheduling methods for both batched and online flows. The evaluations indicate that VLCcube always outperforms Fat-Tree, and our scheduling methods can significantly promote its performance.

ACKNOWLEDGMENTS

The authors thank all the anonymous reviewers for their insightful feedback. Besides, this work is supported by the National 973 Basic Research Program under grant No.2014CB347800, and the National Natural Science Foundation of Outstanding Youth Fund under grant No. 61422214.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM,* Seattle, WA, USA, 2008.
[2] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, and P. Patel, "VL2: a scalable and flexible data center network," in *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009.
[3] H. Navid, Q. Zafar, G. Himanshu, S. Vyas, R. D. Samir, P. L. Jon, S. Himanshu, and T. Ashish, "FireFly: A reconfigurable wireless data center fabric using free-space optics," in *Proc. ACM SIGCOMM*, Helsinki, Finland, 2014.
[4] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, K. Saipriya, V. Amin, B. Y. Zhao and H. Zheng, "Mirror mirror on the ceiling: flexible wireless links for data centers," in *Proc. ACM SIGCOMM,* Helsinki, Finland, 2012.
[5] W. Zhang, X. Zhou, L. Yang, Z. Zhang, B. Y. Zhao and H. Zheng, "3D beam forming for wireless data centers," in *Proc. ACM Hotnets*, Cambridge, MA, USA, 2011.
[6] Y. Cui, H. Wang, X. Cheng, and B. Chen, "Wireless data center networking," *IEEE Wireless Communication*, vol. 18, no. 6, pp. 46–53, 2011.
[7] J.Y. Shin and D. Kirovski, "On the feasibility of complete wireless datacenters," in *Proc. ACM ANCS*, Austin, Texas, USA, 2012.
[8] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Proc. USENIX NSDI,* Lombard, Illinois, USA, 2013.
[9] K. Chen, A. Singla, A. Singh, K. Ramachandran, and L. Xu, "OSA: an optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking,* vol. 22, no. 2, pp. 498–511, 2014.
[10] C. Guo, G. Lu, D. Li, H. Wu, and X. Zhang, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review,* vol. 39, no. 4, pp. 63–74, 2009.
[11] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zheng, and S. Lu, "DCell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM,* Seattle, WA, USA, 2008.
[12] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: a high performance network structure for modular data center interconnection," In *Proc. ACM CoNEXT,* Rome, Italy, 2009.
[13] D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," In *Proc. IEEE ICNP,* Vancouver, BC Canada, 2011.
[14] J. Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proc. ACM SOCC*, Cascais, Portugal, 2011.
[15] A. Shingla, C. Y. Hong, L. Pooa, and P. B. Godfrey, "Jellyfish: networking data centers randomly," in *Proc. USENIX NSDI*, San Jose, Canada, 2012.
[16] L. Gyarmati, and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture", *ACM SIGCOMM Computer Communication Review,* vol. 40, no. 5, pp. 4–12, 2010.
[17] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, and S. Kumar, "Mirror mirror on the ceiling: flexible wireless links for data centers," *ACM SIGCOMM Computer Communication Review,* vol. 42, no. 4, pp. 443–454, 2012.
[18] S. Louvros, D. Fuschelberger, N. Sklavosm, M. Hbner, D. Goehringer, and P. Kitsos, "VLC technology for LTE indoor planning," *System-Level Design Methodologies for Telecommunication*, pp. 21–41, 2014.
[19] D. Tsonev, S. Videv, and H. Hass, "Light Fidelity (Li-Fi): Towards all-optical networking," in *Proc. SPIE Conference Series*, 2013.
[20] D. Tsonev, H. Chun H, S. Rajbhandari, J. Mckendry, S. Videv, E. Gu, M. Haji, S. Mohsin, A. E. Kelly, and G. Faulkner, "A 3-Gb/s single-LED OFDM-based wireless VLC link using a gallium nitride LED," *IEEE Photonics Technology Letters*, vol. pp, no. 7, pp. 637–640, 2014.
[21] Y. C. Chi, D. H. Hsieh, C. T. Tsal, H. Y. Chen, H. C. Kuo, and G. R. Lin, "450-nm GaN laser diode enables high-speed visible light communication with 9-Gbps QAM-OFDM," *Optics Experess*, vol. 23, no.10, 2015.
[22] Ronja, http://ronja.twibright.com/, 2015.
[23] S. Singh and R. Bharti, "163m/10Gbps 4QAM-OFDM visible light communication," *IJETR,* vol. 2, no. 6, pp. 225–228, 2014.
[24] PureLiFi, purelifi.com/lifi-products/li-1st/.
[25] MOMO, axrtek.com/momo/.
[26] TracePro, http://www.lambdares.com/, 2015.
[27] M. Michael, W. R. Andrea, and S. Ramesh, "The power of two random choices: A survey of techniques and results," *Handbook of Randomized Computing*, pp. 255–312, 2000.
[28] K. Han, Z. Hu, J. Luo, and L. Xiang, "RUSH: RoUting and scheduling for hybrid data center networks," in *Proc. IEEE INFOCOM*, Hongkong, 2015.
[29] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, "DCTCP: Efficient Packet Transport for the Commoditized Data Center," *Proc. ACM SIGCOMM*, New Delhi, India ,2010.
[30] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, "TIMELY: RTT-based Congestion Control for the Datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.
[31] Y. Chen, S. Jain, V. K. Adhikari, Z. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! datasets," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011.
[32] D. Abts, M. R. Marty, P. M. Wells, "Energy proportional datacenter networks," *ACM SIGARCH Computer Architecture News,* vol. 38, no. 3, pp. 338-347, 2010.
[33] J. H. Ahn, N. Binkert, A. Davis, "HyperX: topology, routing, and packaging of efficient large-scale networks, " in *Proc. IEEE Conference on High Performance Computing Networking, Storage and Analysis,* Portland, USA, 2009.
[34] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, " Hedera: dynamic flow scheduling for data center networks," in *Proc. USENIX NSDI,* San Jose, CA, USA, 2010.
[35] M. Alizadeh, S. Yang, M. Sharif, S. Katti, and N. Mckeown, "pFabric: minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435-446, 2013.
[36] A. Munir, I. A. Qazi, Z. A. Uzmi, A .Mushtaq, and S. N. Ismail, "Minimizing flow completion times in data centers," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013.