2020

Design and Optimization of VLC Enabled Data Center Network

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

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

Xu Lin  
the School of Computer Science and Technology, Xidian University, Xi’an 710000, China.

Geyao Cheng  
the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China.

Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons

Recommended Citation
Yudong Qin, Deke Guo, Xu Lin et al. Design and Optimization of VLC Enabled Data Center Network. Tsinghua Science and Technology 2020, 25(1): 81-92.

This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in Tsinghua Science and Technology by an authorized editor of Tsinghua University Press: Journals Publishing.
Design and Optimization of VLC Enabled Data Center Network

Yudong Qin, Deke Guo*, Xu Lin, and Geyao Cheng

Abstract: Visible-Light Communication (VLC) has the potential to provide dense and fast connectivity at low cost. In this paper we propose SFNet, a novel VLC-enabled hybrid data center network that extends the design of wireless Data Center Networks (DCNs) into three further dimensions: (1) fully wireless at the inter-rack level; (2) no need for a centralized control mechanism on wireless links; and (3) no need for any infrastructure-level alterations to data centers. Previous proposals typically cannot realize these three rationales simultaneously. To achieve this vision, the proposed SFNet augments fat-tree by organizing all racks into a wireless small-world network via VLC links. The use of VLC links eliminates hierarchical switches and cables in the wireless network, and thus reduces hardware investment and maintenance costs. To fully exploit the benefits of the topology of SFNet, we further propose its topology design and optimization method, routing scheme, and online flow scheduling algorithm. Comprehensive experiments indicate that SFNet exhibits good topological properties and network performance.

Key words: Data Center Network (DCN); Visible Light Communication (VLC); topology design

1 Introduction

Cloud computing is imposing high performance requirements on overlay data centers[1]. Inside each data center, a Data Center Network (DCN) interconnects all of the servers and networking devices to provide massive computing, networking, and storage resources[2]. To improve the network performance of data centers, researchers have proposed many novel DCNs, which can be roughly divided into two categories: wired and wireless. Wired DCNs, such as fat-tree networks[3], connect all of the components with wired links. By contrast, wireless DCNs employ wireless links to interconnect devices, forming either a fully wireless network or a hybrid network using wireless links to augment a wired network.

Existing wired DCNs suffer from several issues. On one hand, they are either over-provisioned with good performance but at a high cost, or oversubscribed at a low cost but with poor performance. Large-scale DCNs are typically constructed based on a hierarchical topology, such as VL2[4] and Portland[5]. In these designs, core switches become the bottleneck for network performance, and can constitute a single point of failure. On the other hand, the cabling process of wired DCNs is inherently difficult and error-prone. To connect the tens of thousands or even hundreds of thousands of nodes inside a data center is both labor-intensive and time-consuming[6]. Besides these connection difficulties, the massive cables also create problems with troubleshooting, reconfiguration, and heat dissipation.

To avoid these problems, researchers have put forward a diverse range of wireless networks to improve on and compensate for wired networks. Wireless communication technology designs construct wireless links to interconnect racks, and also form a wireless network to augment the wired network.
Additionally, there are some radical designs, such as ProjecToR\cite{7} and Firefly\cite{8}, that eliminate wired links altogether and construct fully wireless DCNs at the inter-rack level. Compared with wired links, wireless links are more flexible and can be more easily reconfigured. Thus existing wireless DCNs mainly focus on reconfiguring wireless links based on predicted traffic demands. To achieve this, a powerful but complex control mechanism is required for keeping strict time synchronization, estimating network-scale traffic demand and enabling the centralized reconfiguration of wireless links.

Among current advanced wireless communication techniques, the two most widely used are Free-Space Optics (FSO)\cite{8} and 60 GHz radio frequency communication\cite{9}. Another promising wireless communication is Visible-Light-Communication (VLC), which is also capable of providing high-bandwidth and low-latency connectivity inside data centers. VLC transmits data through visible light, where the absence or presence of light represents “0” and “1”, respectively. It can achieve a 10 Gbps data rate without interference in the communication process. VLCcube\cite{10} first analyzed the feasibility of employing VLC to construct wireless links. Unlike wired links, however, these wireless technologies suffer from a line-of-sight limitation; therefore, existing wireless designs usually impose infrastructure level alterations to data centers to achieve out-of-sight communication.

In this paper, we envision the following three design rationales behind the construction of a high-performance wireless network: (1) wirelessness, constructing a wireless network at the inter-rack level; (2) capacity to plug-and-play, neither needing a centralized control mechanism, nor requiring reconfiguration of the wireless links to match traffic conditions; and (3) easy-deployability, needing no additional infrastructure-level alterations to data centers. Adhering to these rationales can bring profound benefits to data centers. Firstly, hierarchical switches and inter-rack cables are not required when constructing a wireless network, thus reducing hardware investment\cite{11}. Secondly, the configuration and operating procedures of wireless networks can be simplified, with no additional control operations. Thirdly, it does not require alterations to the physical infrastructures inside data centers, for example, changing the ceiling into a mirror. Unfortunately, existing wireless DCNs mainly focus on the flexible reconfiguration of wireless links, and ignore the other two rationales.

To realize these three rationales simultaneously, we propose SFNet, a VLC-enabled DCN. It augments fat-tree, a representative production wired DCN, by organizing all racks into a wireless small-world network via VLC technology. Hence, SFNet is a hybrid DCN, which seamlessly integrates the wired fat-tree and wireless small-world networks together. The major contributions of this paper are summarized as follows:

- We design SFNet, a hybrid data center network. It employs VLC wireless links to interconnect all of the racks in a fat-tree data center into a wireless small-world network. The topology design and optimization strategies ensure high bandwidth and low average path length. SFNet simultaneously achieves each of the three wireless network rationales listed above.
- To fully exploit the benefits of the topology of SFNet, we design a hybrid routing scheme to jointly utilize both wireless and wired links. To further improve the network performance, we propose a congestion-aware scheduling method to optimize the online flow scheduling problem.
- Trace-driven experiments are conducted to evaluate the performance of SFNet under different traffic patterns. The results show that SFNet does achieve the expected performance.

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 reports on how to construct and optimize SFNet. Section 4 puts forward the routing method for SFNet, and proposes the congestion-aware online flow scheduling method. Section 5 evaluates the performance of SFNet in both its topological properties and network performance. Section 6 concludes this paper.

2 Related Work

Wired data centers suffer from severe cabling complexity, which is inherently difficult and error-prone. In these networks, the large number of cables result in design and development problems related to wire ducting and maintenance, as well as high operation costs (amounting to about 7% – 8% of the total infrastructure cost)\cite{12}. Moreover, cable bundles in wired DCNs can block airflow, reducing cooling efficiency and increasing energy consumption.

To solve these intrinsic problems of wired DCNs, researches have introduced many wireless
communication technologies to construct wireless links. The most widely used wireless communication technologies are 60 GHz and FSO. Flyway first analyzed the feasibility of employing 60 GHz to connect racks. However, both 60 GHz and FSO suffer from a line-of-sight limitation; therefore, researchers have proposed many methods to enable out-of-sight communication. 3D beamforming and Firefly employ 60 GHz and FSO respectively, to construct wireless links. To avoid blocking on the 2D plane, both 3D beamforming and Firefly use a huge ceiling mirror to bounce signals. To implement this method, the space above racks needs to be completely clear, and there are strict requirements for the height of the ceiling mirror. Unfortunately, this method is not feasible in most data centers because of their complex steel structure and the air conditioning pipes above the racks. Diamond places reflectors of wireless signals beside racks forming 3D ring reflection spaces for server-to-server connections. However, the enclosed 3D ring around racks makes cooling and maintenance difficult. ProjectToR abandons these public reflective designs, instead placing a micro mirror and mirror assembly on each rack for private use, which requires high precision in installation and configuration.

Existing DCNs employing wireless links mainly focus on flexibility. In these designs, the wireless links can be reconfigured according to the current or predicted traffic demand. However, to permit such reconfigurations, a data center requires a complex control mechanism to keep strict time synchronization, estimate the network-scale traffic demand, and enable centralized assign link configuration. In this paper, we adopt the widely used fat-tree as an example of a wired DCN and augment it with a wireless network. In this way, we interconnect all Top of Rack (ToR) switches according to a dedicated structure by VLC links, and achieve a hybrid SFNet that can seamlessly integrate both wired and wireless DCNs.

Recent analysis of Facebook traffic reveals that the communication inside a data center is extremely widespread. In a fat-tree network, traffic needs to traverse up and down the network hierarchy, thus exacerbating the suboptimal throughput. By contrast, random topologies are proven to give optimal throughput for uniform traffic. For the purpose of compensating for this fat-tree limitation, we utilize VLC links to interconnect ToR switches into a small-world structure, a random topology that is resilient to failures and miswirings.

Based on the above considerations, we finally design SFNet, which seamlessly integrates the wired fat-tree and wireless small-world networks. The size of a fat-tree is related to the number of ports on switches. Let \( k \) denote the number of ports of each switch. The SFNet then consists of \( k \) pods, each of which has \( k/2 \) ToR switches. Thus, there are \( k^2/2 \) racks in total. For all racks in the fat-tree, VLC links are employed to form a wireless small-world network. In SFNet, the VLC links forming a small-world network can be classified into two categories: regular and random. The regular links interconnect the racks into a wireless grid topology. Since visible light is a line-of-sight communication, racks have to be placed in a grid, with \( m \) racks in each row and \( n \) racks in each column. Transceivers of regular links are installed on the side of each rack, while transceivers of random links are installed on the top. The hierarchical layout of VLC links is illustrated in Fig. 1.

**3 Design of SFNet**

### 3.1 Topology design of SFNet

Inside a data center, racks are usually interconnected through upper level network devices instead of connecting with each other directly, thus forming a hierarchical network structure. Clos, or multi-rooted tree, is the de-facto standard network architecture for data centers because of its easy implementation. In this paper, we adopt the widely used fat-tree as an example of a wired DCN and augment it with a wireless network. In this way, we interconnect all Top of Rack (ToR) switches according to a dedicated structure by VLC links, and achieve a hybrid SFNet that can seamlessly integrate both wired and wireless DCNs.

Recent analysis of Facebook traffic reveals that the communication inside a data center is extremely widespread. In a fat-tree network, traffic needs to traverse up and down the network hierarchy, thus exacerbating the suboptimal throughput. By contrast, random topologies are proven to give optimal throughput for uniform traffic. For the purpose of compensating for this fat-tree limitation, we utilize VLC links to interconnect ToR switches into a small-world structure, a random topology that is resilient to failures and miswirings.

Based on the above considerations, we finally design SFNet, which seamlessly integrates the wired fat-tree and wireless small-world networks. The size of a fat-tree is related to the number of ports on switches. Let \( k \) denote the number of ports of each switch. The SFNet then consists of \( k \) pods, each of which has \( k/2 \) ToR switches. Thus, there are \( k^2/2 \) racks in total. For all racks in the fat-tree, VLC links are employed to form a wireless small-world network. In SFNet, the VLC links forming a small-world network can be classified into two categories: regular and random. The regular links interconnect the racks into a wireless grid topology. Since visible light is a line-of-sight communication, racks have to be placed in a grid, with \( m \) racks in each row and \( n \) racks in each column. Transceivers of regular links are installed on the side of each rack, while transceivers of random links are installed on the top. The hierarchical layout of VLC links is illustrated in Fig. 1.

![Hierarchical layout of VLC links](image-url)
On average, we install 4 transceivers for each rack, thus the number of VLC links is the same as a torus of the same size. As shown in Fig. 1a, racks inside the grid hold 4 transceivers to construct regular links with their adjacent racks, while racks on the edge of the grid can construct no more than 3 regular links. The spare transceivers are used to construct random links, of which they can form \( m+n \) in total. Thus some racks can have more than 4 transceivers installed. Figure 1b is an example of random links.

Figure 2 shows a complete topology of SFNet resulting from the above construction method. SFNet is a hybrid data center network. The ToR switches are connected as a fat-tree using wired links, and also as a wireless small-world network using VLC links.

### 3.2 Construction and spatial layout of random links

The SFNet integrates the wired fat-tree and the wireless small-world networks seamlessly. The construction method of fat-tree networks can be found in the literature\[3\]. In this paper, we focus on constructing random links in the wireless small-world and addressing some challenges with deployment.

#### 3.2.1 Construction of random links

As described above, VLC links are introduced to construct a small-world network to augment the wired fat-tree network. In a data center, racks are connected into a grid with VLC links. To complete a small-world network, random links are added to the grid as shortcuts between two nodes, which can reduce the network latency. For the construction of random links, the simplest method is to select two racks completely at random and connect them with VLC links. However, data centers can perform better if the random links are constructed based on small-world phenomenon.

In the construction of SFNet, Kleinberg’s small-world model\[20\] is adopted to promote the routing efficiency.

In Kleinberg’s small-world model, random links are chosen with the probability proportional to the \( d \)-th power of the corresponding distance (i.e., the shortest path length) between any two racks. Here, \( d \) refers to the number of dimensions of the lattice network. Since the random links are constructed on a grid in our VLC-based small-world network, the value of \( d \) is 2. To measure the shortest distance between two racks, we need information about the coordinates in the grid. Based on the properties of the grid, we can easily obtain the distance between two racks as follows:

\[
d(x, y) = |x_i - y_i| + |x_j - y_j|
\]

where \( x \) and \( y \) represent two racks with coordinates \( (x_i, x_j) \) and \( (y_i, y_j) \), respectively. For example, if the coordinate of the first rack is \((3, 4)\), while that of the second rack is \((2, 1)\). Then their distance is calculated as \( d = |3-2|+|4-1| = 4 \).

In a realistic deployment, each pair of racks is assigned a value proportional to the square of their distance. The value is normalized to the interval \((0,1)\), thus it can be used as the probability that the pair of racks are connected with a random link. In this way, we can construct random links in accordance with the small-world theory.

#### 3.2.2 Spatial layout of random links

In SFNet, the wireless VLC links constitute a small-world network, and they can be classified into regular links and random links. The regular links connect adjacent racks and form a grid network, while the random links mainly connect racks across different pods. This increases the connectivity of racks at the pod level. The introduction of wireless VLC links eliminates the cables among racks, thus avoiding cabling problems.

However, using VLC links to connect racks does present some challenges. Firstly, the VLC links use visible light to transmit data, which has a line-of-sight limitation. Communication will be distributed if the light paths of VLC links are blocked by any other devices or obstacles; it is therefore crucial to ensure that there is nothing positioned on any light path. Secondly, differently from lasers, a VLC link generated from an LED is not an “ideal thin line”. The beam width of each VLC link may exceed the required receive-range. As the transceivers have great sensitivity\[21\], they may
receive signals from light paths which are close to them and cannot distinguish these interference signals with their own signals.

As described in our design rationales, we do not intend to introduce any devices, such as mirrors, to relay VLC signals. Thus it is necessary to ensure that the two transceivers of a VLC link can fully face each other and not be blocked by any infrastructure component. To solve the blockage and interference problems, we must deploy the transceivers of regular links and random links hierarchically so as to separate all VLC links in space. Specifically, the transceivers of regular links are installed on the side of racks, while transceivers of random links are installed on the top of racks. The reasoning behind this is that regular links are only constructed between adjacent racks, while random links often cross over many racks. On each side of a rack, we install one transceiver, and two connected transceivers should be at the same height.

For random links, transceivers are installed on the top of racks to avoid interference with regular links. However, if these transceivers are installed at the same height, they may block each other. To address this layout problem of random links, we transform this problem into a vertex coloring problem. Typically, the vertex coloring problem is to assign “colors” to vertices of a graph while ensuring that no edge connects two vertices with the same color. For random links, we first need to construct the interference graph $G^{[14]}$, which describes the interference relationship of the random links. Each vertex in $G$ corresponds to a random link, while each edge in $G$ corresponds to an interference. If two random links intersect, then their corresponding vertices in $G$ will be linked together.

Based on the interference graph $G$, we can use the vertex coloring theory to assign colors to random links. Let $l$ represent the number of required colors. Random links can then be divided into $l$ groups based on the assigned colors, and only random links with the same color are installed at the same height. Moreover, if the location of the transceivers is flexible, we can also adjust their location horizontally to avoid some intersections, thus decreasing the value of $l$.

### 3.3 Optimization of SFNet

To fully utilize the benefits of VLC links, we optimize the topology of SFNet. More precisely, the placement of racks and the construction of random links are optimized to improve the performance of SFNet.

#### 3.3.1 Optimization of rack placement

To construct SFNet, we introduce VLC links to construct a small-world network. For a data center with $m \times n$ racks, the VLC-based small-world network comprises an $m \times n$ grid and $m + n$ random links. As a data center network, its performance is related to its Average Path Length (APL). We can decrease the APL by optimizing the placement of racks.

For an $m \times n$ grid, its APL $d_P$ can be calculated as follows:

$$d_P = \sum_{j=1}^{n} \sum_{i=1}^{m} \frac{1}{m^2 n^2} \left[ \frac{i(j - 1)m}{2} + \frac{(n - j + 1)(n - j)m}{2} + \frac{i(i - 1)n}{2} + \frac{(m - i + 1)(m - i)n}{2} \right] = \frac{1}{3} \left( m + n - \frac{1}{m} \frac{1}{n} \right)$$

Equation (2) depicts the relation between APL and network size. Accordingly, we can achieve the minimal $d_P$ in the case of $m = n$ when the network size is fixed. Given the number of racks in a data center, we should place racks in an $n \times n$ square to optimize the average path length of the VLC-based grid.

Through the optimization of rack placement, the APL of grid can be decreased. Note that, the APL of an $n \times n$ grid increases with $n$ linearly, leading to poor scalability. Also, the diameter of an $n \times n$ grid is $2n - 2$, while that of a torus topology with the same size is only $n$. The average path length in a grid is also longer than that in a torus. Thus, it is necessary to add some random links to improve the performance of a data center network.

#### 3.3.2 Optimization of random links

In a small-world network, links can be classified into two types: the regular links forming the grid, and the random links for decreasing the average path length. To construct SFNet, we install an average of 4 transceivers on each rack. Apart from those transceivers for constructing the grid, the remaining transceivers can still form $2n$ random links. Thus the number of VLC links in the small-world network is the same as that in an equally sized torus.

Each VLC link requires two transceivers on different racks. To construct random links easily, we can construct a small-world network independently. However, since SFNet is a hybrid network and the small-world network is constructed based on a fat-tree,
we can optimize the construction of random links by incorporating the properties of the fat-tree topology.

In fat-tree, racks are organized as pods. The path length of any two racks in the same pod is 2 hops, while that of two racks in different pods is 4 hops. By adding one random link between two racks inside a pod or across two pods, the path length will decrease by 1 hop or 3 hops, respectively. Obviously, for the purpose of shortening the average path length it is more effective to add random links between racks in different pods than in the same pod. Additionally, having many random links connect the same pair of pods should be avoided, otherwise the benefits of adding random links will be weakened due to the marginal effect.

To clarify the presentation, we first assign two-part identifiers to each rack. Suppose that each switch in the fat-tree network has \( k \) ports. The prefix of an identifier ranges from 0 to \( k-1 \), and denotes the pod this rack belongs to. The suffix ranges from 0 to \( k^2-1 \), and identifies the rack in each pod. For example, the identifier 32 refers to the third rack in the fourth pod.

To measure the effect of adding random links, we define the pod graph and pod connectivity as follows:

*Definition 1* Given the topology of a data center network, the pod graph is an undirected graph \( G=(V, E) \). Each vertex \( v \in V \) corresponds to a pod in the data center. Each edge \( e \in E \) between vertices \( v_1 \) and \( v_2 \) means that there exist random links connecting the corresponding pods in the data center.

*Definition 2* For a pod graph \( G=(V, E) \), the pod connectivity is defined as the ratio of the edge number in \( G \) to that in the corresponding complete graph. The pod connectivity of \( G \) can be calculated as follows:

\[
C_G = \frac{2|E|}{(|V|)(|V|-1)}
\]  

Pod connectivity can be no greater than 1. Figure 3 depicts an example with \( k=6 \). Figure 3a is the topology of the VLC-based small-world network with a pod graph as illustrated in Fig. 3b. The pod graph consists of 6 vertices and 11 edges; hence, the pod connectivity is 0.73 as calculated with Eq. (3).

To make the pod connectivity as large as possible, we repeat the process of generating random links many times and obtain the corresponding random link scheme. For each random link scheme, we calculate its pod graph and pod connectivity. The scheme with the maximal pod connectivity will be then adopted. The construction process of SFNet is simple but efficient. Moreover, fat-tree has the reputation of being easily deployable[18], and the construction of a small-world network is also simple, meaning that our hybrid SFNet is also easy to deploy.

4 Routing and Congestion Aware Flow Scheduling in SFNet

In SFNet, wired links coexist with wireless links. Thus we can find three kinds of routing paths between two racks: wired paths, wireless paths, and hybrid paths. In this section, we first explore the routing scheme in SFNet. To minimize network congestion and balance the traffic load, we formulate a congestion-aware flow scheduling model and design a scheduling algorithm for online traffic patterns.

4.1 Routing scheme

The routing algorithm for a wired path under the fat-tree structure can be found in Ref. [3]. This paper explores the routing algorithms for wireless and hybrid paths.

4.1.1 Greedy routing with wireless links

Obviously, transmitting packets along the shortest path consumes the least time. However, finding the shortest path requires high computation complexity. If the shortest paths between all rack pairs are precomputed, each rack has to maintain and store a routing table, the size of which is proportional to the scale of the data center.

To efficiently route under the small-world structure, we design a greedy routing algorithm that achieves both low complexity and acceptable path length. In this algorithm, each rack determines the next hop based on the distance from its neighbors to the destination rack. The distance is computed with Eq. (1). Suppose the source rack and destination rack are \( r_c \) and \( r_d \), respectively. The current rack is \( r_c \) and the greedy routing path is \( P \). The steps of the greedy routing
The identifiers assigned in Section 3.3.2 are used again to examine the hybrid path that can shorten the wired path. One of these two racks is an end of this VLC link. In the SFNet, the length of a wired path can be shortened by a hybrid path when a VLC link connects the two pods, and the hops between two racks in a fat-tree structure are the same pod or different pods, respectively. Therefore, to design an efficient routing algorithm for the hybrid path.

As in the case of routing with wireless links, generating links seamlessly together and generate hybrid routing paths. Two kinds of links coexist in SFNet: wired links and wireless links. We have described the routing methods of link are not fragmented; instead, they can work on the same screen. The hops between two racks in a fat-tree structure are reduced path length and increased storage cost is optimal when \( h = 3 \).

4.1.2 Hybrid routing method

Two kinds of links coexist in SFNet: wired links and wireless links. We have described the routing methods using a single type of link. However, the two kinds of link are not fragmented; instead, they can work seamlessly together and generate hybrid routing paths. As in the case of routing with wireless links, generating the shortest hybrid path between two racks incurs high computation complexity. Thus, it is still necessary to design an efficient routing algorithm for the hybrid path.

The hops between two racks in a fat-tree structure are either 2 or 4, depending on whether two racks belong to the same pod or different pods, respectively. Therefore, the length of a wired path can be shortened by a hybrid path only when a VLC link connects the two pods, and one of these two racks is an end of this VLC link. In this case, the path length can be reduced to 3 hops.

Based on the above analysis, we prefer only to examine the hybrid path that can shorten the wired path. The identifiers assigned in Section 3.3.2 are used again to present the hybrid routing method. Suppose the identifiers of two racks are \( xy \) and \( uw \). The routing steps are as follows:

Step 1: For the two racks \( xy \) and \( uw \), judge whether there is a VLC link on rack \( xy \) connecting with pod \( u \), or a VLC link on rack \( uv \) connecting with pod \( x \). If not, the routing process is terminated.

Step 2: Select a VLC link connecting rack \( xy \) (or \( uv \)) with pod \( u \) (or \( x \)). Without loss of generality, suppose this link is \((xy, uw)\), which connects rack \( xy \) and pod \( u \). The hybrid path can then be denoted as \((xy, uw, S_u, uv)\), where \( S_u \) is any aggregation switch in pod \( u \), and \((uw, S_u, uv)\) is the wired path from \( uw \) to \( uv \).

For example, the wired path length between racks 11 and 50 is 4, and there is a VLC link connecting rack 11 with rack 52, which is located in the same pod as rack 50. In such a case, we can construct a hybrid path. Compared with the wired path, the hybrid path does not traverse any core switch. This algorithm is also efficient since it only searches the VLC links on two racks and avoids unnecessary computation.

4.2 Problem formulation of flow scheduling

The SFNet has great path diversity, including wired path, wireless path, and hybrid path. To use links efficiently and decrease the degree of congestion, we formulate the flow schedule problem. 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 \( c(e) \). Additionally, \( F=\{f_1, f_2, \ldots, f_\delta\} \) represents \( \delta \) injected flows in \( G \). For each flow, \( f_i=(s_i, d_i, b_i) \), \( s_i \) and \( d_i \) denote its source node and destination node, respectively, while \( b_i \) is its traffic demand. The variable \( \xi_i(e) \) represents whether flow \( f_i \) passes through link \( e \). The value of \( \xi_i(e) \) is 1 if \( e \) is passed by \( f_i \), otherwise it is 0. Let \( \phi \) record the schedule strategy of \( F \) in \( G \).

**Definition 3** Given \( F \) and \( \phi \), we define the congestion rate of link \( e \) as

\[
C^{\phi}(e) = \sum_{i=1}^{\delta} \xi_i(e) \cdot b_i / c(e) \tag{4}
\]

Note that any \( C^{\phi}(e) \) falls into a constant interval \([0, 1]\). Specifically, if no flow passes through link \( e \), its utilization rate is 0. The utilization rate is 1 when link \( e \) is fully used.

**Definition 4** We define the congestion rate of a path \( P \) as

\[
C^{\phi}(P) = \max_{e \in P} C^{\phi}(e), \quad e \in P \tag{5}
\]

The congestion rate reflects the congestion condition. For any path, we accordingly locate the bottleneck and decide whether the path is capable of supporting a given flow.

With the above definitions, the flow scheduling problem can be formulated as follows:

\[
\text{Minimize } \sum_{e \in E} C^{\phi}_{\text{inf}}(e)^2,
\]

\[
\sum_{i=1}^{\delta} \xi_i(e) \cdot b_i \leq c(e), \quad \forall e \in E \tag{6}
\]

\[
\sum_{e \in \text{out}(s_i)} \xi_i(e) - \sum_{e \in \text{in}(s_i)} \xi_i(e) = 1, \quad \forall i \tag{7}
\]
The flows which need to be scheduled are then represented as $F_S = F_N + F_R$. We can define the online flow scheduling problem as follows:

**Definition 5** The OFS problem is to update the schedule strategy with the least increase in link congestion. Let $\Delta C = C_S - C_0$, where $C_S = \max C_{F_S}(e)$ and $C_0 = \max C_{F-FR}(e)$. The goal of OFS is to minimize the $\Delta C$.

Since $C_0$ is determined, to minimize the $\Delta C$ is equal to minimizing the $C_S$. For any flow $f_i \in F_S$, there are three alternative paths: 1 hybrid path, 1 wireless path, and $k^2/4$ wired paths. The set of these candidate paths of $f_i$ is represented as $\mathcal{P}(f_i)$. Our method is described in Algorithm 1. It first finds the flows needing to be scheduled (Lines 1, 2). Then, for each flow, we find all of its candidate paths and the corresponding path congestion rates (Lines 3, 6). The path with the least congestion rate will be selected as the flow scheduling strategy (Lines 7, 8).

**5 Performance Evaluation**

In this section, we evaluate the performance of SFNet in two aspects: qualitative analysis and quantitative comparisons. For the latter, we first introduce the experiment settings and methodologies, then we compare SFNet with VLCcube in terms of topological properties and network performance. Finally, the online flow scheduling method is compared with the widely used Equal-Cost Multi-Path (ECMP).

**5.1 Qualitative comparison**

In the process of qualitative analysis, we compare 4 DCNs: SFNet, Firefly, 3D BeamForming (3D BF), and ProjecToR. Firstly, the communication technologies they use to construct wireless links are VLC, laser, 60 GHz and laser, respectively. Among these, VLC has the cheapest cost and shortest range. Secondly, all of the DCNs except for SFNet, are flexible, since their wireless links can be reconfigured based on traffic demand. Thus they need to maintain a complex control plane. The difference between them is that the control planes of Firefly and 3D BF are centralized, while that of ProjecToR is distributed. In ProjecToR, each switch reconfigures wireless links only based on local traffic. By contrast, SFNet is a static DCN; its links are fixed and stable. Therefore, it does not need to impose control on wireless links. Thirdly, SFNet introduces no infrastructure-level alterations to data centers. Firefly and 3D BF introduce reflective ceilings, and ProjecToR installs a mirror array and mirror ball for each rack. Fourthly, SFNet is plug-and-play. Once the links are configured, no extra control and coordination...
mechanism are needed. In contrast, the other three DCNs design their wireless links based on complicated mechanical or electrical control operations. Moreover, they need to predict the traffic demand accurately for reconfiguring wireless links, which is time-consuming and costly.

Therefore, SFNet can simultaneously achieve three design rationales, i.e., wirelessness, ease of deployment, and capacity to plug-and-play. It offers a stable topology and an easy routing mechanism at the cost of giving up slightly on the requirement of flexibility.

5.2 Setting and methodology of evaluation

To evaluate the topological properties and network performance of SFNet, we compare SFNet with VLCcube. Just like SFNet, VLCcube is a hybrid DCN which augments a wired fat-tree network with a VLC-based wireless network. At the same scale, VLCcube and SFNet have an equal number of VLC links. The main difference is that the VLC links in VLCcube form a torus network while those in SFNet form a small-world network.

In our simulation experiments, we use NS3 to realize VLCcube and SFNet. Given \( k \) (the number of ports per switch), we generate the VLCcube according to the rules described in Ref. [10], and the SFNet is generated as we describe in this paper. The bandwidth of both wired links and VLC links are set to 10 Gbps. The link delay is set to 1 ms in accordance with the literature[23, 24]. With these settings, we compare SFNet with VLCcube in terms of topological properties, including the routing complexity of the three kinds of paths, and network performance.

To evaluate the network performance under both uniform and non-uniform traffic, we consider the following two traffic patterns introduced in the literature[25]:

Random Uniform (RU) traffic: In this pattern, the source rack and destination rack of a flow are generated randomly, thus flows are distributed evenly across the whole network.

Random Non-uniform (RN) traffic: In this pattern, 75 percent of flows are distributed evenly across the whole network, and the remaining 25 percent of flows are distributed among the first 12.5 percent of racks. In RN, traffic is unbalanced, and some hot regions may be generated.

To prove the effectiveness of our OFS method, we evaluate the performance of SFNet under both ECMP and OFS. The arrival time of online flows follows a Poisson distribution.

5.3 Topological properties

We first compare the average path length of the wireless networks in SFNet (small-world) and VLCcube (torus). The results are shown in Fig. 4a. As we noted above, the same sized SFNet and VLCcube have an equal number of VLC links. However, the average path length of the small-world structure is 31% less than tours on average, and the gap between them reaches 62% when \( k = 30 \). Figure 4b shows the average path length of the complete SFNet and VLCcube. The average path length of SFNet is 6.1% shorter than VLCcube. Both SFNet and VLCcube have much shorter average path length than their wireless network, since each path in the fat-tree is no more than 4 in length.

Figure 4c reflects the routing complexity of the three kinds of paths (wireless, wired, and hybrid). Routing for wireless paths takes the most time, while routing for wired paths takes the least and most stable amount of time.

We conduct the generation process multiple times to deduce the random links of the small-world, and the solution with the highest pod connectivity is adopted. In Fig. 4d, SFNet1, SFNet2, and SFNet10 denote the resulting pod connectivity after 1 rounds, 2 rounds, and 10 rounds, respectively. Obviously, the pod connectivity increases with the number of rounds, and decreases with the network scale.

5.4 Network performance

In this section, we evaluate the network performance of
SFNet and VLCcube with metrics of throughput and latency. Latency is depicted as the average finish time of all flows, while throughput is normalized against the real throughput of VLCcube. In the experiments, we use two traffic patterns (RU and RN) and two flow sizes (4 MB and 12 MB). The flow scheduling method is the widely used ECMP. For each situation, we vary $k$ from 6 to 30.

### 5.4.1 Network performance under RU traffic

In the random uniform traffic pattern, we inject $k^3$ flows into SFNet and VLCcube to evaluate their network performance. For each flow, its source rack and destination rack are selected randomly, so that flows spread evenly across the whole network.

Figure 5 demonstrates the throughput under different sizes of flows. With a flow size of 4 MB and 12 MB, SFNet achieves 1.2 and 1.1 times the throughput of VLCcube on average. Figure 6 depicts the latency under different sizes of flows, and shows that SFNet causes less latency than VLCcube. Based on these experiment results, we can say that SFNet achieves better performance than VLCcube when the traffic is spread evenly across the whole network.

### 5.4.2 Network performance under RN traffic

In the random non-uniform traffic pattern, we also inject $k^3$ flows into SFNet and VLCcube to evaluate their network performance, this time under unbalanced traffic. Specifically, the first 12.5% of racks bear 25% of the flows, and the remaining 75% of flows are uniformly distributed across the whole network.

Figures 7 and 8 demonstrate the throughput and latency, respectively, under the RN traffic pattern. Compared with the results under the RU traffic pattern, both SFNet and VLCcube perform more poorly under the RN traffic pattern. However, SFNet still outperforms VLCcube, achieving 1.09 and 1.12 times the throughput of VLCcube with flow sizes of 4 MB and 12 MB. The latency of SFNet is also lower than that of VLCcube. Thus we can say that SFNet performs better than VLCcube under unbalanced traffic.

### 5.4.3 Performance of OFS method

The above experiments are simulated with the ECMP method. To fully exploit the topological benefits of SFNet, we design the Online Flow Scheduling (OFS) method based on the properties of SFNet. Comparisons between OFS and ECMP are made to evaluate the performance of OFS, with throughput and packet loss rate measured under different network scales.

We inject $k^3$ random flows into SFNet, with $k$ varying from 6 to 30. The arrival time of dynamic flows follows a Poisson distribution. The results are shown in Fig. 9, form which we can see that OFS outperforms ECMP in terms of both throughput and packet loss rate. Throughput is normalized against the real throughput with ECMP, showing that, on average, OFS can achieve 1.3 times the throughput of ECMP. The packet loss rate of OFS is less than that of ECMP, and it almost reduces to 0 when $k=18$. It is clear that our OFS strategy promotes the performance of SFNet significantly. The main reason is that OFS can provide more candidate
paths, and spreads the flows as widely as possible in SFNet.

6 Conclusion

In this paper, we propose SFNet, an easily-deployable and high-performance hybrid DCN architecture. SFNet introduces VLC links to construct a wireless small-world network to augment the wired fat-tree network. These VLC links decrease the average path length and improve network performance. To solve the line-of-sight limitation of VLC links, we propose a spatial layout method for transceivers to avoid blockage of VLC links. To fully exploit the topological benefits of SFNet, we design a dedicated flow scheduling method for online flows. The evaluations indicate that SFNet outperforms VLCcube, and our flow scheduling method promotes its performance.

References

[1] D. Guo, J. Xie, X. Zhou, and X. Zhu, Exploiting efficient and scalable shuffle transfers in future data center networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 997–1009, 2015.

[2] Z. Ding, D. Guo, X. Liu, X. Luo, and G. Chen, A MapReduce-supported network structure for data centers, *Concurrency & Computation Practice & Experience*, vol. 24, no. 12, pp. 1271–1295, 2012.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, A scalable, commodity data center network architecture, *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[4] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, V12: A scalable and flexible data center network, *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

[5] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, Portland: A scalable fault-tolerant layer 2 data center network fabric, *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39–50, 2009.

[6] Y. Cui, H. Wang, X. Cheng, and B. Chen, Wireless data center networking, *IEEE Wireless Communications*, vol. 18, no. 6, pp. 46–53, 2011.

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

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

[9] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, Augmenting data center networks with multi-gigabit wireless links, *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 38–49, 2011.

[10] L. Luo, D. Guo, J. Wu, S. Rajbhandari, T. Chen, and X. Luo, VLCube: A VLC enabled hybrid network structure for data centers, *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2088–2102, 2017.

[11] D. Guo, C. Li, J. Wu, and X. Zhou, DCube: A family of network structures for containerized data centers using dual-port servers, *Computer Communications*, vol. 53, pp. 13–25, 2014.

[12] H. Vardhan, S.-R. Ryu, B. Banerjee, and R. Prakash, 60GHz wireless links in data center networks, *Computer Networks*, no. 58, pp. 192–205, 2014.

[13] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, Mirror mirror on the ceiling: Flexible wireless links for data centers, *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 443–454, 2012.

[14] Y. Cui, S. Xiao, X. Wang, Z. Yang, S. Yan, C. Zhu, X. Li, and N. Ge, Diamond: Nesting the data center network with wireless rings in 3-d space, *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 145–160, 2018.

[15] 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. the Conference of the ACM Special Interest Group on Data Communication*, Los Angeles, CA, USA, 2017, pp. 267–280.

[16] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, Inside the social network’s (datacenter) network, *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 123–137, 2015.

[17] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, and A. Kanagala, Jupiter rising: A decade of clos topologies and centralized control in Google’s datacenter network, *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 183–197, 2015.

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

[19] A. Singla, C. Y. Hong, L. Popa, and P. B. Godfrey, Jellyfish: Networking data centers randomly, in *Proc. NSDI*, San Jose, CA, USA, 2012, pp. 225–238.

[20] J. Kleinberg, The small-world phenomenon: An
algorithmic perspective, in *Proc. the ACM Symposium on Theory of Computing*, Montreal, Canada, 2002, pp. 163–170.

[21] S. Vijay and K. Geetha, A survey on visible light communication appliances used in inter-vehicular and indoor communication, *International Journal of Applied Engineering Research*, vol. 11, no. 7, pp. 4893–4897, 2016.

[22] K. Han, Z. Hu, J. Luo, and L. Xiang, Rush: Routing and scheduling for hybrid data center networks. in *Proc. INFOCOM*, Hong Kong, China, 2015, pp. 415–423.

[23] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, and B. Prabhakar, Data center TCP (DCTCP), *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 63–74, 2010.

[24] R. Mittal, V. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, TIMELY: RTT-based congestion control for the datacenter, *ACM SIGCOMM Computer Communication Review*, 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 Transactions on Parallel and Distributed Systems*, vol. 21, no. 12, pp. 1765–1778, 2010.

**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 the 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, Beijing, China, in 2001, and the PhD degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a professor with the College of System Engineering, National University of Defense Technology, and is also with the College of Intelligence and Computing, Tianjin University. 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.

**Xu Lin** received the BS degree from Xidian University, Xi’an, China, in 2016. He is currently a master student in Xidian University. His research interests include software-defined network, data center network, edge computing, and network function virtualization.

**Geyao Cheng** received the BS degree from National University of Defense Technology, Changsha, China, in 2017. She is currently working towards the MS degree in the College of Systems Engineering, National University of Defense Technology, Changsha, China. Her research interests include data centers and software-defined networks.