Thor: A Server-level Hybrid Switching Data Center Network With Heterogeneous Topologies

Xiaoshan Yu*
City University of Hong Kong
Xidian University
China
xsyu@xidian.edu.cn

Hong Xu
City University of Hong Kong
Hong Kong
henry.xu@cityu.edu.hk

Huaxi Gu
Xidian University
China
hxgu@xidian.edu.cn

ABSTRACT

Optical interconnects are promising to address the dilemma faced by traditional electrical networks in data centers. However, some issues, including weak connectivity of the topology, limited scalability of the control system, and inefficient use of WDM technologies, limit the deployment potential of optical networks. To address these issues, we propose a novel hybrid network called Thor. By employing different topologies for the optical and electrical networks, Thor achieves server-level optical interconnects and can better exploit the unique properties of circuit switching and packet switching. To build a more scalable and responsive control plane, Thor uses a distributed control system built atop the electrical network. Moreover, we develop a new optical switch to more efficiently utilize multiple wavelengths. Our evaluation results show that Thor is able to deliver ∼ 90% bisection bandwidth of a non-blocking electrical network under realistic traffic patterns. Moreover, by transmitting mice and elephant flows separately in each of the two networks, Thor is able to provide an order of magnitude speedup in average flow completion times for mice flows compared to conventional approaches.

CCS CONCEPTS

• **Networks → Data center networks**;

KEYWORDS

Data center network, optical network, hybrid switching, topology

1 INTRODUCTION

Data Center Networks connect hundreds of thousands of servers and support various applications. To cope with the stringent performance requirements, many new network architectures have been proposed in recent years. Electrical networks using the commodity Ethernet switches, such as fat-tree[1], VL2[2], BCube[3], and DCell[4], have been deployed in practice. These topologies are primarily designed to provide full-bisection bandwidth. They are not particularly energy efficient: in practice the network is rarely full utilized, but electrical switches consume the same amount of power even without traffic.

To provide higher bandwidth with lower power consumption, the networking community then proposes to employ optical interconnects in data centers. Hybrid optical/electrical network architectures, such as Helios[5], C-through[6], and Mordia[7], and all-optical network designs, such as OSA[8], DOS[9], and WaveCube[10], have been studied. Their deployment potential, however, are fundamentally limited by the following issues.

First, most existing designs only provide optical access at the rack level by connecting optical switches to ToR (Top of Rack) switches. This is mainly due to the practical constraint of very low port density of optical switches. However, the limited fan-in/fan-out reduces the number of concurrent optical paths that can be established. Traffic may be frequently bottlenecked at the ToR tier, especially considering that one-to-many and many-to-one transmission patterns are common in applications with partition-aggregate workflows. Additionally, the large number of electrical ToR switches undermines the energy saving of optical components[11].

Second, in existing work, the forwarding decisions and the corresponding optical path configuration are managed in a centralized manner by a dedicated controller. Although a centralized control plane is easy to implement, it may suffer from scalability and robustness issues especially for large-scale networks. Moreover, an additional out-of-band control network is needed to connect the controller to all optical devices and the ToR switches.

Third, WDM (Wavelength Division Multiplexing) is a crucial technology to increase the network capacity and overcome the high blocking ratio in optical networks. However, limited by the optical switch structure, most existing designs cannot exploit multiple wavelengths with high cost- and power-efficiency. The space-based optical switches, such as MEMS (Micro-Electro-Mechnical-Systerm) and SOA (Semiconductor Optical Amplifier), are wavelength transparent. The wavelength-based optical switches, such as WSS(Wavelengths Selective Switch) and AWGR (Arrayed Wavelength Grating Router) are more expensive and power hungry[12].

Limited by above issues, the optical connections can only be deployed in the core layer, acting as a complementary network. With the development of silicon photonic devices, the switching speed has been reduced from milliseconds to microseconds[13]. The optical circuit can be operated at much finer granularity. Now it is make sense to offload more traffic to optical network. Thus we present THOR, a server-level hybrid switching architecture with heterogeneous topologies. It solves the connectivity limitations by employing a more scalable topology for optical network. Then it

*The work is done when Xiaoshan was in City University of Hong Kong.

ACM TUR-C '17, May 12-14, 2017, Shanghai, China
© 2017 ACM. ISBN 978-1-4503-4873-7/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3063955.3063992

eliminates the centralized control plane by designing a distributed control system. Moreover, THOR develops a more efficient way to use multiple wavelengths by designing a new optical switch structure.

In summary, the main contributions of this paper are as follows:

- We propose Thor, a server-level hybrid switching network to achieve the scalability, high performance and low power consumption. Thor provides optical connections directly to hundreds of thousands of servers by employing the hypercube topology. Thus THOR can achieve lower power consumption compared to electrical and prior hybrid network designs. We also develop a distributed control system using the electrical network to support the fast optical circuit switching (OCS).
- We design a new optical switch structure combining the WSS and MEMS modules to realize the wavelength-based multi-hop optical communication. Moreover, this optical switch is able to operate with multiple wavelengths in a more cost-efficient way: the cost of the optical switch does not increase with the number of used wavelengths. Moreover, much energy is saved by reducing the number of switching modules and eliminating the power-hungry optical devices such as the optical/electrical converters and tunable wavelength converters (TWC). Further by exploiting the properties of this multi-wavelength switch, an optical path-shared routing algorithm is developed to reduce the blocking ratio of the optical circuit network.
- We present the performance evaluation of Thor using realistic simulations. Our results demonstrate that Thor is able to achieve nearly 90% of the nonblocking bandwidth, which is much better than the prior optical circuit networks. As to the average flow completion time for mice flows, Thor achieves an order of magnitude of performance improvement by separating the elephant and mice flows into two networks.

2 TOPOLOGIES

Thor is a hybrid network architecture with server-level optical links as shown in Fig. 1. Each server has two NICs and can access both networks directly. The optical network adopts a low-radix hypercube topology, while the electrical network uses a high-radix flattened butterfly topology to better exploit the characteristics of different switching technologies. The electrical network also serves as the control plane for the optical network.

In the following, we explain the motivation for and design of the heterogeneous topologies for the electrical and optical networks.

2.1 The optical network topology: Hypercube

As the microsecond optical circuit switching becomes available, it is feasible to provide server-level optical connections since in this case link utilization approaches 80% even if the optical connection only carries a single flow larger than 100KB.

Designing the topology with server-level optical circuits is more challenging than existing hybrid DCN designs. On one hand the topology should be scalable to interconnect a huge number of servers. On the other hand, current fast optical switches have very low port density. For example, the typical form of a commercial 2D-MEMS switch with microsecond switching time is 4×4 or 8×8. Thus a low-radix topology with a longer diameter is a natural choice. This tradeoff is acceptable because of the following reasons: first, in data center the optical circuits are mainly responsible for delivering large flows, which take large fraction (80%−90%) of data volume but are rare ($< 10\%$) in number[14]. Thus in practice the probability of optical connection collision is very small once we provide enough access capability. This will lead to a low delay for path setup. Second, in circuit network the communication delay is dominated by the path setup time rather than the number of hops between the source and destination.Thus the diameter of the topology is not the key factor determining transmission delay. Third, the elephant flows are insensitive to delay, it is affordable to take time to establish a multi-hop optical path.

Some low-radix but highly scalable topologies, such as mesh, torus, and hypercube, can be chosen as the candidates for optical network. However, as optical circuits use the communication channels exclusively, a topology with higher path diversity can potentially provide higher connectivity. In general, hypercube provides more edge-disjoint paths than mesh or torus when connecting the same number of servers. Thus Thor adopts hypercube to establish the optical network. It directly connect tens of thousands of servers using optical switches with dozens of ports. It uses 2^n optical switches to form an n-dimensional hypercube as shown in the right part of Fig. 1 (with $n = 5$). Each optical switch has $2n$ ports. Half of the ports are connected to the servers while the remaining ports are connected to it adjacent switches in n dimensions. An n-dimensional hypercube connects $n \times 2^n$ servers in total.

2.2 Electrical Network Topology: Flattened Butterfly

The electrical network carries mice flows that are vast in number but only represent a small fraction (5%–10%) of the total bytes [2, 15]. Thus it is built using the low bandwidth and inexpensive Ethernet switches to reduce the cost. Different from optical switches, the port density of electrical switches is large especially at low data rates. High radix topologies, such as butterfly, folded Clos, flattened butterfly, and dragonfly, can be considered to build the electrical network. Thor prefers to use flattened butterfly (FB) because of the following two reasons: first, leveraging the high-radix switches, FB achieves lower average distance (i.e. delay), wiring complexity, and power consumption than folded Clos [16]. While it provides higher inter-group connectivity than dragonfly. Second, unlike the indirect topologies such as butterfly and folded Clos, FB has servers attached to every switch directly. By utilizing these processing nodes, each switch has the extra computing power to execute complicated control instructions. This greatly facilitates us to integrate the control plane into the electrical network.

Specifically, Thor's electrical network uses 1GbE switches to form a k-ary m-flat flattened butterfly (FB) with m dimensions. In each dimension, a switch uses $(k - 1)$ ports to connect to every other switch in the same dimension, and it connects to $m \times (k - 1)$ switches in total. In addition, each switch also connects to c servers. Fig. 1 shows a 16-ary 1-flat FB, with 16 switches in each dimension and each switch connecting to 15 other switches and 10 servers.

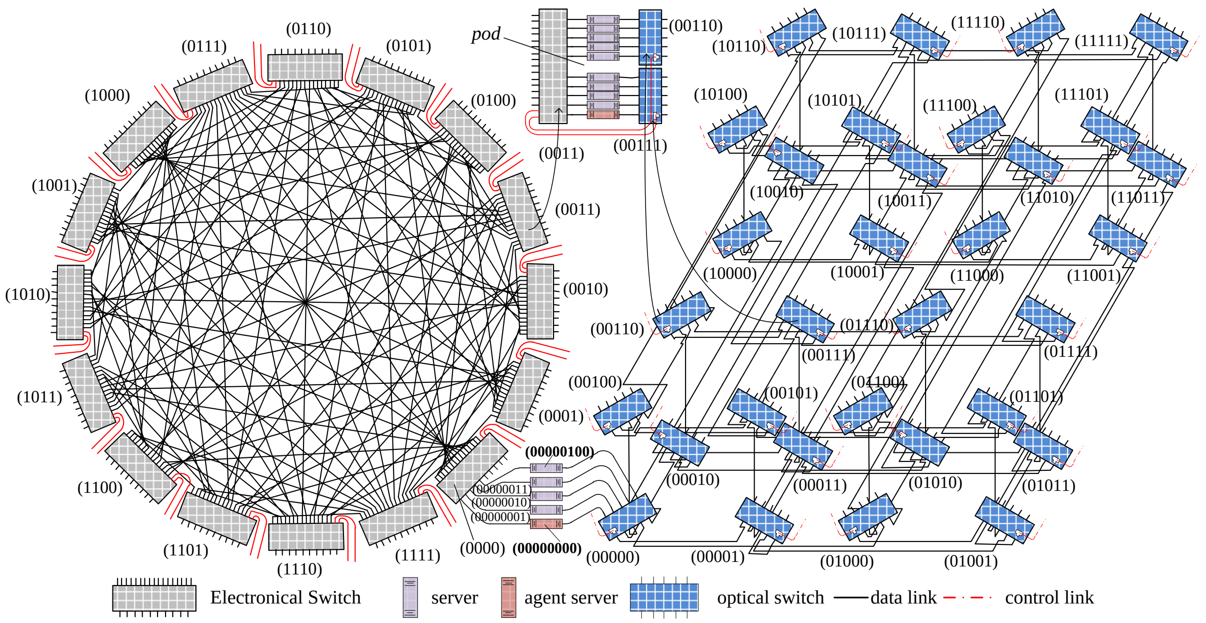


Figure 1: The architecture of Thor

In general, a k-ary m-flat FB with c servers per switch can be expressed as FB(c, k, m). The parameters c, k, and m can be configured based on the traffic conditions of the network. Thor's design requires that they satisfy the following:

$$c \cdot k^m = n \cdot 2^n, \qquad c, k, m, n \in \mathbb{Z}, \tag{1}$$

$$2^n / k^m = 2^p, \qquad k, m, n, p \in \mathbb{Z}. \tag{2}$$

Equation 1 ensures FB hosts the same number of servers as the optical hypercube topology. Equation (2) ensures that there is a good correspondence between the electrical switch and the optical switch. This relationship facilitates us to embed the optical control system into the electrical network and makes it easy to identify the position of a device in both the optical network and the electrical network.

2.3 Addressing

Like DCell and BCube, Thor also uses a set of customized addressing rules to identify the network devices. Specifically, binary sequences are used as the addresses of the switches and servers. Based on the dimension of the hypercube n, each optical switch is addressed as $(a_{n-1}a_{n-2} \ldots a_i \ldots a_0)$, where $a_i \in \{0, 1\}$. Based on this addressing rule, two optical switches are connected through the $(n+i)^{th}$ port if their i^{th} bit are different while others are the same. In the electrical network, a switch is given a $(n-p)$-bit binary sequence $(e_{n-p-1}e_{n-p-2} \ldots e_0)$. Based on the dimensions of FB topology, this address is divided into m segments. If the addresses of two electrical switch only differ in one segment, they are connected. The n-dimension hypercube connects $n \times 2^n$ servers, each addressed with a $(n+s)$-bit $(2^s \geqslant n)$ binary sequence $(b_{n+s-1}b_{n+s-2} \cdots b_0)$. The first n bits $b_{n+s-1}b_{n+s-2} \ldots b_s$ follow the address of the optical switch it connects to while the last s bits $b_{s-1}b_{s-2} \ldots b_0$ denote the position of the server within the local optical switch.

2.4 Control Plane

Different from prior work, Thor employs a distributed control system to manage the optical network, performing functions including the optical path setup, conflict resolution, optical switch configuration, etc. Thor embeds the control system into the electrical network to avoid the need of an extra control network. As shown in Fig. 1, one electrical switch, 2^p optical switches, and $n \times 2^p$ servers form a basic building block named *pod* ($p = 1$ and $n = 5$ in Fig. 1). The electrical switch connects to each console port of the 2^p optical switches in the same pod. One server in the pod acts as the control agent, and can configure the switching state based on the control information. Further, to setup an optical path traveling through multiple pods, the agent servers in these pods can communicate to make coordinated configurations.

2.5 Scalability Analysis

The radix of optical switch becomes an important consideration in determining the scalability of Thor because the low-speed Ethernet switch is able to provide a large number of switching ports. i.e. the port count of Cisco Catalyst 6500 Series can scale from 16 ports up to 576 ports in a single chassis. While a 50×50 MEMS-actuated silicon photonic switch with $2.4us$ switching time has been reported in [17], it is not yet commercially available. In [7], a 24×24-port fast optical switch is built from commercial 1×4 switching units. Thus the fast optical switching module with port count less than 20 can be expected to realize in today's market. On the other hand, as will be explained in the next section, in Thor the n-port optical switch is actually built from $n/2$-port switching modules. for example, a 15 dimensional hypercube capable of connecting 491,520 servers is constructed by 15-port optical switching modules. Therefore, Thor holds the promise of scaling beyond 100,000 servers in practice.

As the main objective of THOR is to greatly enhance the capacity of optical interconnects, this architecture needs to reserve some ports for the future expansion. However, the optical portion does not need to reserve too many ports because it inherently uses low-radix switches. i.e. Only reserving 8 ports, the optical network of THOR can scale from 2048 servers up to 49,152 servers. As to the electrical portion, the Ethernet switches can use the pluggable modular switch infrastructure [18] to reduce the reservation power and cost.

3 THE WAVELENGTH ADD-DROP SWITCH

WDM is commonly used to reduce the high blocking ratio of the optical network. However, currently there is no cost- and energy-efficient optical switch to support the multiple wavelengths switching. The space-based optical switches such as MEMS [5, 6] and SOA [19] cannot conduct fine-grained operations on each wavelength (i.e. when a WDM signal enters one input port of the space-based switch, it cannot switch different wavelengths to different output ports). Although the wavelength-based switches AWGR [9] and WSS[10] are able to switch any wavelengths between an arbitrary pair of input-output ports, they have limitations either in power-consumption or scalability. Specifically, AWGR requires the energy-hungry optical element TWC to route the optical signal to the desired output port. WSS on the other hand has very low port density in practice (1×9 and 1×20 are commercially available). Thus we need to design a more energy-efficient and low-cost multi-wavelength optical switch using commodity optical modules.

Typically in multi-hop optical circuit switching network, a WDM switch capable of operating Φ wavelengths needs Φ MEMS modules. Although this switch is able to forward any wavelength signal between an arbitrary pair of input-output ports, the cost of the design is high: If we introduce α wavelengths to the network, the total cost increases α times at least compared to the signal-wavelength optical network.

To find a more cost-efficient way to introduce the WDM technique. We re-analyze the connection collisions in multi-hop optical networks. Two connections will collide with each other if a subset of the links are overlapping and they use the same wavelength. These collisions can be classified into two types. The first type is shown in Fig. 2(a), where two connections share the same links from the first collision node to the destination switch of one connection. In the second type of collisions, shown in Fig. 2(b), the two connections deflect to different paths after sharing the same links. We observe that the required wavelength operation is different to handle these two types of collisions. A full wavelength-switching function which is able to switch any wavelength from any input to any output is required at intermediate switches to resolve the second type of collisions. However, for the first type of collisions, the intermediate switch just needs the function of adding a wavelength from the local ports (the ports connected to the server) to the inter-switch ports (the ports connected to other switches) and vice versa. Thus at the cost of giving up full wavelength-switching function, we design a wavelength add-drop switch to achieve a more cost-efficient way of using multiple wavelengths.

As shown in Fig. 3, the main switching fabric of our switch consists of n 1×2 WSS modules, three n×n MEMS modules, and n 2×

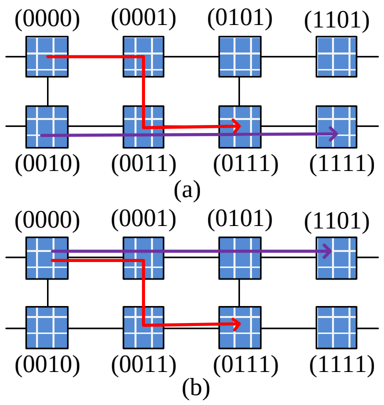


Figure 2: The collision of two optical connections.

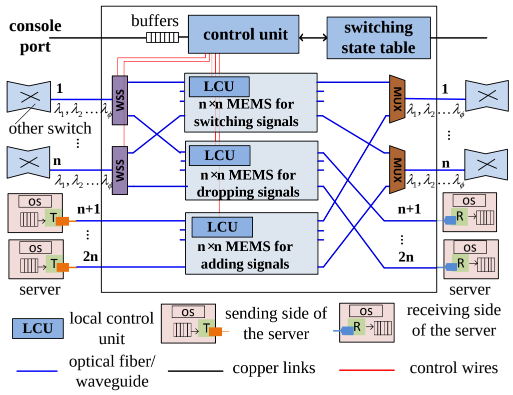

console port
buffers
control unit
switching state table
1
$\lambda_1, \lambda_2 \dots \lambda_\phi$
other switch
n
$\lambda_1, \lambda_2 \dots \lambda_\phi$
WSS
WSS
LCU
n×n MEMS for switching signals
LCU
n×n MEMS for dropping signals
LCU
n×n MEMS for adding signals
MUX
MUX
1
$\lambda_1, \lambda_2 \dots \lambda_\phi$
n
$\lambda_1, \lambda_2 \dots \lambda_\phi$
OS
T
n+1
OS
T
2n
server
OS
R
n+1
OS
R
2n
server
LCU
local control unit
OS
T
sending side of the server
OS
R
receiving side of the server
optical fiber/ waveguide
copper links
control wires

Figure 3: The structure of the wavelength add-drop switch.

1 multiplexers. The three MEMS modules have different functions. The first module is responsible for switching optical signals between different inter-switch ports. The second module is responsible for switching signals from network to the local servers, while the third module is responsible for sending signals from the local servers to other switches. The 1×2 WSS module is equipped at each input of the inter-switch port, with one of its output ports connected to the first MEMS module and the other connected to the second module. When a WDM signal arrives at the input port, one single-wavelength signal destined to the local servers is separated by the WSS and switched to the destination by the second MEMS module. The rest wavelengths of the WDM signal have the same output port and these wavelengths are delivered by the first MEMS module. The optical signal sent from the local servers is firstly delivered to the third MEMS module and then switched to the required output port. Finally at the output port this signal is assembled with other signals from the first MEMS module.

This wavelength add-drop switch provides better price to wavelength ratio than multi-wavelength switches. Moreover, because the multiple wavelengths are often switched together in this switch, these signals in fact share the energy of driving one mirror into on-state. Thus this switch also consumes less energy than the multi-wavelength switch.

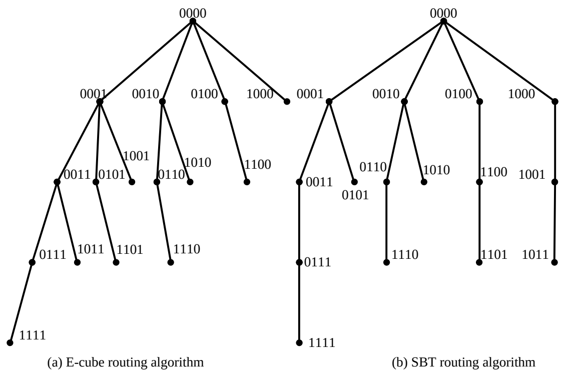


Figure 4: The connections from the same source but but based on the different routing algorithm

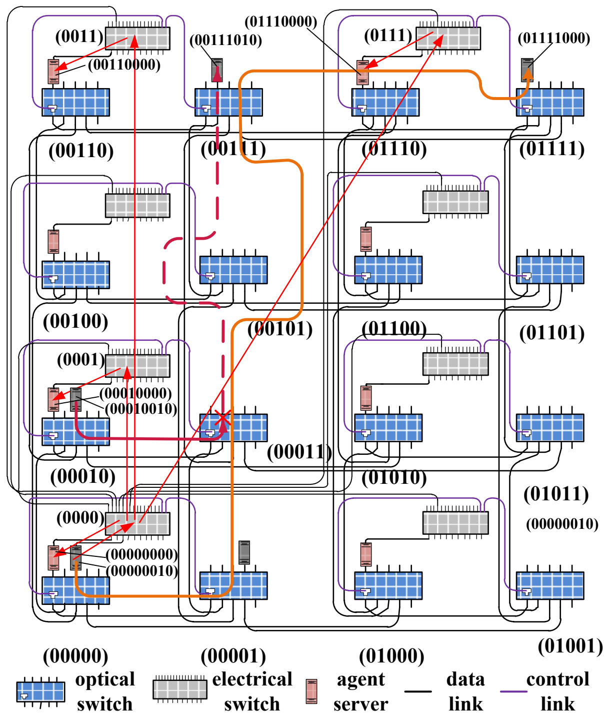


Figure 5: A example of optical path setup

4 ROUTING

As a hybrid network, Thor uses packet switching for mice flows and circuit switching for elephant flows. We assume that host based elephant detection such as Mahout [20] is in place. Alternatively, we can use the following simple strategy to differentiate mice and elephants: each flow is regarded as a mice flow initially and sent to the electrical network. If the amount of the data it sends has reached a threshold, the flow is recognized as an elephant and the source host initiates the process of establishing an optical path for it.

We now explain the routing algorithms for both mice and elephant flows, as well as the optical path setup mechanism.

4.1 Routing for mice flows

Mice flows are transmitted via the electrical network in the store-and-forward manner. A packet is firstly buffered in the queue. Then the scheduler calculates the output port based on the destination address of the packet and sends it to the next hop. As THOR topology and connection rules are deterministic for the operators, the shortest path routing algorithm proposed in [21] can be adopted to determine the output port. Specifically, the scheduler firstly extracts the address of the switch to which the destination server is connected (call destination E-switch). Then it compared this address with its own address from the lowest segment to the highest segment. (As stated in section 2.3, the address of an electrical switch has m segments, with each indicating its position in one dimension of the FB topology.) If two addresses differ in the i^{th} segment, the scheduler then sends this packet to the next switch which has the same i^{th} segment as the destination E-switch.

4.2 Routing for elephant flows

THOR uses hypercube for the optical fabric. Thus an optical path needs to be established over several optical switches. Multi-hop circuit switching is adopted to make coordinated configurations of the corresponding switches. When a server detects an elephant flow, it sends the setup packet to reserve the wavelength resources on the optical switches along the path. If one of the intermediate switches has leased its resource to other connects, this connection is blocked. The optical connections are easily blocked if there is only one wavelength. Thus using multiple wavelength is necessary to reduce the blocking ratio.

THOR relies on the wavelength add-drop switches introduced in Section 3 to support multi-wavelength communication. Only connections with completely overlapped links from the first collision switch to the destination of one connection can share the path using different wavelengths. Thus the routing algorithm directly affects the number of path-sharing connections.

We adopt the Spanning Balanced Tree (SBT) algorithm [22] for elephant flow routing here. There are two reasons: First, SBT uses the shortest path and minimizes the number of links used and potential collisions. Second, as shown in Fig. 4, in hypercube we can use a tree to describe all connections from the same source: the root node represents the source while the leaf nodes represent the destinations. Based on two shortest-path routing algorithms – e-cube [23] and SBT, the corresponding trees are generated as shown in Fig. 4(a) and (b). The tree generated by SBT has fewer branches. Thus SBT enables more connections to share a path by using wavelength add-drop switches.

To reduce the processing delay, the source server calculates the complete transmission path based on SBT routing algorithm and packages this information into the setup packets. When a agent server receives a setup packet, it directly extracts the path information and begins the path setup procedure which is described in the following subsection.

4.3 Optical path setup

The delay of the optical path setup is crucial to the link utilization. This delay is actually determined by both the hardware switching time and software control time. Based on the recently proposed photonic components, the hardware switching time has been reduced from milliseconds to microseconds. In this case, software control time becomes the dominant component for the setup delay. In practice, it is challenging for a centralized control system to finish the control loop within tens of microseconds especially with hundreds of switches. To solve this problem, we firstly develop a distributed control system to reduce the number of switches each controller manages. Furthermore, we propose a fast control strategy to establish multiple optical paths in parallel.

We firstly use an example to intuitively illustrate the procedure of path setup. Fig 5 shows a partial network which is extracted from Fig. 1. Here the optical switches form a 5-dimensional hypercube topology while the electrical switches forms a 16-ary 1-flat FB topology. To make the background syllabify, we omit the links between the electrical switches. One electrical switch, two optical switches, and 10 servers build a pod. In this pod, one server is designated as the control agent. By sending instruction to the console ports, the agent server is able to configure the switching state of the optical switches in the same pod. Based on this control system, a server can selfishly establish an optical path. For example, in Fig 5 server (00000010) wants to build an optical path to server (01111000). It firstly calculates the transmission path based on SBT algorithm. Then it gets the specific optical switches which needs be configured for this path. These switches are (00000), (00001), (00011), (00111), and (01111). The source server further finds the agent servers managing the required optical switches, which are server (00000000), (00010000), (00110000), and (01110000). Then through the electrical network, the source server sends path setup requests to every agent server. When receiving this request, the agent begins to configure the corresponding optical switches and returns the acknowledge packet to the source. After the source server collects all response from the four agent server. It knows the optical path has been established and begins the data transmission. Finally, the source server informs the four agent of the path releasing.

We now describe the detailed procedure of path setup. At the beginning the source of the elephant flow randomly selects a wavelength λ_r to use. (Here r represents the index of the wavelength. Assume THOR employs Φ wavelengths and they are labeled as $\lambda_0, \cdots,$ $\lambda_r, \cdots, \lambda_{\Phi-1}$). Then it determines the complete transmission path for the flow using SBT routing algorithm. In a n-dimensional hypercube, assume there are h-hops between the source and destination servers, then the transmission path can be described using the addresses of all switches, i.e. $(\chi_{[n\cdot(h-1)-1]}\chi_{[n\cdot(h-1)-2]}\cdots\chi_0)$, where $(\chi_{n\cdot i-1}\chi_{n\cdot i-2}\cdots\chi_{n\cdot(i-1)})$ is the address of the i^{th} hop switch. For each intermediate optical switch, the source server also determines its corresponding agent servers responsible for controlling it. As we have already defined the special relationship between the optical and electrical topologies, it is easy to locate the agent server for an optical switch. For example, if the address of an optical switch is $(\alpha_{n-1}\alpha_{n-2}\cdots\alpha_0)$, then the address of its agent server has the form of $(\alpha_{n-1}\alpha_{n-2}\cdots\alpha_p 00\cdots 0)$, where the first $(n-p)$ bits are extracted from the address of the managed optical switch and the last $(s+p)$ bits are all 0. After finding the agent server, the source begins to send path setup requests. Suppose all $(h-1)$ optical switches are managed by ρ agent servers (h is the hop count between the source and destination server), the source server then generates ρ setup packets each destined to one agent server to deliver information about the wavelength λ_r and the path. Note setup packets are transmitted in the electrical network. When an agent server receives a setup packet, it firstly finds the required output port in its corresponding optical switches. Based on these information, the agent server further checks whether there are other connections on the require ports. If no, it records the path information, sends the configuration commands to the corresponding optical switches, and returns an ACK packet to the source server. If other connections are using the required output ports, the agent server checks each collision port and determines if the required connection can share the port with existing ones. If both of the following two conditions hold, path-sharing is permitted:

Condition 1: The new connection uses a different wavelength with the existing ones;

Condition 2: It is possible to share the port as determined by Algorithm 1.

The basic idea of the Algorithm 1 in **Condition 2** is to check whether the required connection has the completely overlapped path with the existing connections. Since in the existing connections, the one with the maximum hops will overlap with all other connections, as shown in line 2 of Algorithm 1, the longest connection is selected to compare with the required connection. Then in line 3-10, if the remainder links of the two connections, from the collided switch to the destination of one connection, are all same, the path-sharing is permitted. Otherwise, the path-sharing is forbidden. As this algorithm only involves simple operations such as searching and string comparison, the operation for path sharing achieves a constant time complexity, which ensures low computation overhead and processing delay.

Furthermore, if all collision ports of the managed optical switches allow sharing for the new connection, the agent server configures the corresponding optical switches, records the path information and makes the response to the source server. Otherwise, the agent server temporarily buffers the setup packet and waits for the release of the corresponding resources.

When the source server collects ρ ACK packets, an optical path has been established. Then the source server begins to transmit the elephant flow through the optical network. After finishing the transmission, ρ tear-down packets are sent to the agent servers to release the reserved wavelength. Note we use the high-radix Ethernet switches to reduce the diameter of the electrical network and reserve the extra bandwidth for the control packets, the control information can be delivered quickly in the electrical network.

Note in above path setup scheme, the reserved ports and links can only be released by the tear-down packets. This rule eliminates the illegal state change and ensures the correct end-to-end transmission. For example, assume in Fig. 5 server (00010010) requires to establish an optical path (named $Path2$) to server (00111010) using wavelength λ_γ. By this time, an optical path (named $Path1$)

Algorithm 1 The algorithm for path sharing judgment

```
Input: The list which records the existing connections on the specific port: $\{L^{port\_id}_{switch\_id}\}$; the path information carried by the setup packet:($\chi_{[n\cdot(h+1)-1]}\chi_{[n\cdot(h+1)-2]}\cdots\chi_0$); The conflicted switch with the required input and output ports: $\{(\beta_{n-1}\beta_{n-2}\cdot\beta_0)(P'_{in}, P'_{out})\}$
Output: $F_{share}$
 1: finding the list $L^{P'_{out}}_{\beta_{n-1}\beta_{n-2}\dots\beta_0}$, which records the information of the existing connections at the output $P'_{out}$ of the switch ($\beta_{n-1}\beta_{n-2}\dots\beta_0$)
 2: finding the longest binary sequence ($\mu_(n\cdot\kappa-1)\mu_(n\cdot\kappa-2)\dots\mu_0$) from all the recorded path information
 3: Extracting a subsequence $\Gamma$ from ($\chi_{[n\cdot(h+1)-1]}\chi_{[n\cdot(h+1)-2]}\cdots\chi_0$), which only contains the addresses of the optical switches on the remainder path from current switch to the destination switch. $\Gamma$ = ($\gamma_{n\cdot\tau-1}\gamma_{n\cdot\tau-2}\dots\gamma_0$)
 4: $F_{share} = True$
 5: for ($i = 0$; $i < min(n\cdot\kappa, n\cdot\tau)$;$i++$) do
 6:     if ($\gamma_i \neq \mu_i$) then $F_{share} = False$ and break
 7:     end if
 8: end for
 9: if ($F_{share} = True$) then adding the path information $\Gamma$ into the list $L^{P'_{out}}_{\beta_{n-1}\beta_{n-2}\dots\beta_0}$
10: else buffering the arrived setup packet and waiting for the resource releasing
11: end if
```

using the same wavelength has been set up from server (00000010) to (01111000). These two connections will collide at switch (00011). However, source server (00010010) does not perceive this collision. It sends three setup packets for requiring the wavelength channels at switch (00010), (00011), and (00111) respectively. Switch (00010) and (00111) configure corresponding resource and return the ACK packets. However, switch (00011) dose not respond to the source server because of the collision. Then the source will wait for the last ACK packet. During this period, switch (00010) and (00111) will keep the configuration state unchanged because they do not receive any control packets from the source server. After *Path*1 finishes the transmission and sends tear-down packets, switch (00011) will return the last ACK packet at once. At this time, *Path*2 is established completely and the source server begins the data transmission.

5 EVALUATION

In this section, we evaluate Thor from the following aspects. First, we calculate the cost and power consumption of Thor and compare it with state-of-the-art electrical and optical networks. Then we analyze the network performance of Thor via packet-level simulations.

5.1 Cost and power

The cost and power consumption for different data center networks is estimated based on the values listed in Table 1. We choose fat-tree, BCube, and Helios, as the baseline network architectures corresponding to switch-centric, server-centric, and optical/electrical hybrid interconnects, respectively. As 10Gbps copper cables can only support short-distance transmissions, the optical transceivers and fibers are employed as the cross-rack links in the electrical network. Moreover, to quantify the maximum cost and power consumption, in most cases of our evaluation the electrical portion of Thor is configured to be an over-provisioned network with $c/k < 1$.

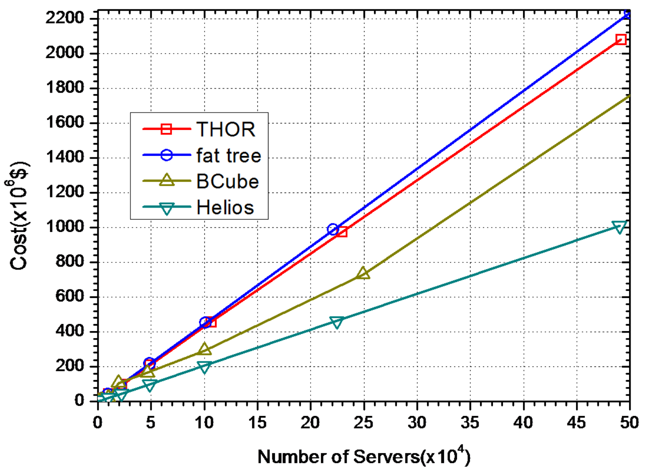
Table 1: cost and power for different devices

Device	Cost ($)	Power (W)
1 Gbps Electrical Network Interface Card [24]	50	1.9
10 Gbps Optical Network Interface Card (with wavelength tunable transmitter)[11]	155	5.5
10 Gbps Electrical Network Interface Card[11]	140	13.4
Wavelength Tunable Filter[11]	50	0.2
1Gbps Electrical Switch[11]	10†	3.5 †
10Gbps Electrical Switch [25, 26]	450†	9.5 †
MEMS Switch[25, 27]	450†	0.24 †
1Gbps Copper Cable[25]	10	0
Fiber Cable[25]	25	0
WSS[10]	1000 †	1 †
(DE)Mux[10]	3000	0
Couple[10]	100	0
10Gbps Transceiver[27]	400	1

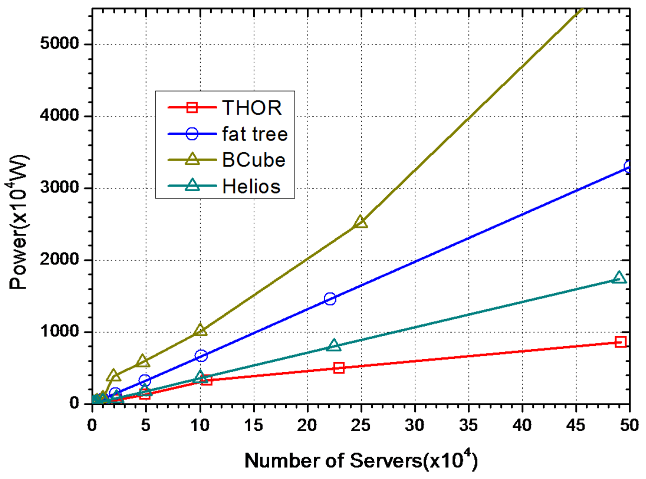
∗ The symbol † means the cost or power is per port value

The results are shown in Fig. 6. It can be observed that Thor improves the power efficiency significantly while maintaining an acceptable capital investment. To be more specific, Fig. 6(a) depicts the cost of different networks as the size varies from thousands to hundreds of thousands of servers. In contrary to the common belief, Thor's optical network does not suffer from high cost. With the same number of servers, Thor reduces the cost by ∼5% compared to fat-tree. The main reason is that to deliver inter-rack traffic, most ports of the electrical switch require additional optical transceivers. This makes the per-port price of the electrical switch higher than that of the MEMS switch. By shifting part of packet forwarding to servers, BCube uses fewer switching ports and therefore has lower cost compared to Thor and fat-tree. Helios achieves the lowest cost by deploying the MEMS switches at the core layer and avoiding using other more expensive optical devices such as WSS. Although Thor is more costly than Helios, it provides richer connectivity and better switching capacity. Moreover we believe Thor can enjoy more cost savings in future because of the following two reasons. First, the prices of WSS and MUX (multiplexer) will drop once they become widely adopted in data centers. Second, if we upgrade the link bandwidth to 40Gbps, all electrical networks including fat-tree and BCube need to deploy more expensive electrical switches and transceivers, while Thor only needs to upgrade the network interface cards since the optical switches are bit-rate transparent.

Fig. 6(b) shows the power consumption of networks with different scales. It is evident that Thor greatly improves the power consumption of the network. Specifically, it consumes 53%–70% less power than fat-tree, 69%–80% less than BCube, and 23%–44% less than Helios. Overall, Thor cuts the power consumption of the electrical network by over half. This result is expected for the following two reasons. First, the electrical switches consume more



(a) Cost consumption comparison.



(b) Power consumption comparison.

Figure 6: Cost and power consumption comparison of different data center networks.

energy than optical ones at 10Gbps. Second, additional power is introduced when packets need to make O/E and E/O conversions at each hop. Compared to the traditional hybrid network Helios, Thor removes the energy-hungry edge switches or ToR switches by extending the optical connection to servers. This makes it possible to further reduce the power consumption of the hybrid networks.

5.2 Performance evaluation

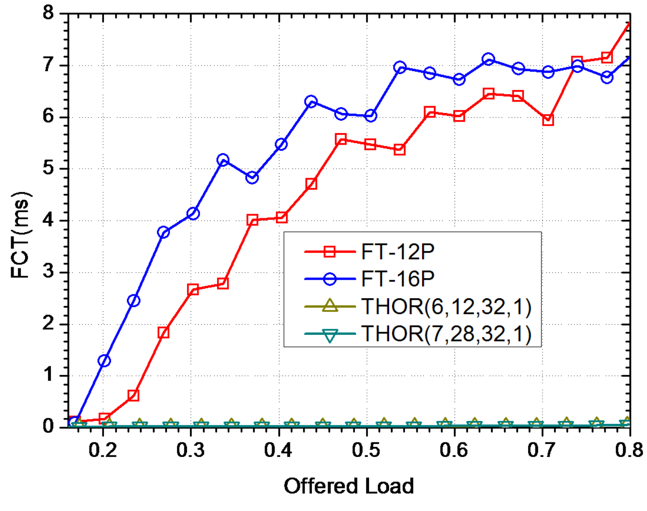
We evaluate the performance of Thor using extensive packet-level simulation based on the OPNET simulator. In the simulations, two Thor networks with different sizes are used. The first one is called Thor(6, 12, 32, 1) which hosts 384 servers with a 6-dimensional hypercube for the optical network and FB(12, 32, 1) for the electrical network. The other network is Thor(7, 28, 32, 1) which hosts 896 servers with a 7-dimensional hypercube and a FB(28, 32, 1). 12-pod and 16-pod fat-tree networks with 432 and 1,024 servers, respectively, are used as the comparison baseline. For Thor, the bandwidth of the copper links and electrical switches is set to 1Gbps, while the optical network uses 80 wavelengths with each having 10Gbps bandwidth. For fat-tree, link bandwidth is set to 10Gbps and flow-based ECMP is used as the multi-path routing algorithm.

The flow size in our simulation follows a heavy-tailed distribution which is abstracted from a data center with the majority workload being data mining jobs[2]. Specifically, 80% of the flows being smaller than 10KB and other flows varying from 100kB to tens of MB. Packet size of a mice flow is set to 200B while the packet size of an elephant flow is 1500B based on [28]. Flows are generated following a Poisson process with varying mean arrival rates to simulate different offered loads to the network. The source and destination of a flow is generated randomly. We consider two performance metrics: flow completion time (FCT), and end-to-end delay experienced by a packet.

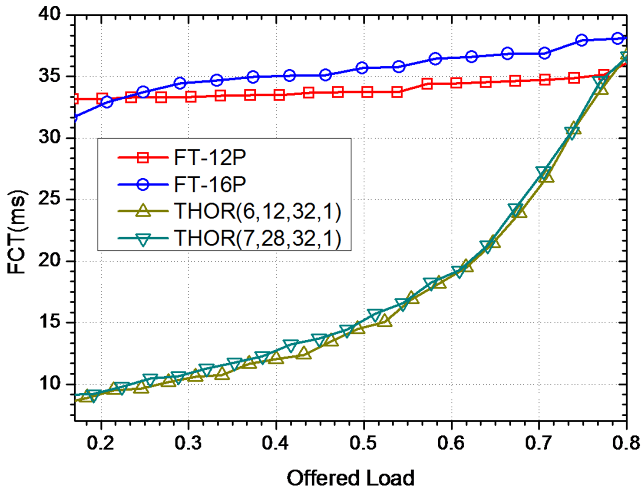
We firstly evaluate the FCT of Thor compared to fat-tree. As shown in Fig. 7(a), Fig. 7(b), and Fig. 7(c), Thor improves FCT significantly. For mice flows, Thor reduces the mean FCT by almost an order of magnitude compared to fat-tree as in Fig. 7(a). Quantitatively, the mean FCT of mice flows in Thor varies from about 0.012 *ms* to 0.2 *ms* as the traffic load increases from 0.1 to 0.8, while that in fat tree varies from 0.1 *ms* to 7.9 *ms* as the traffic load increases. This is reasonable as it has been observed that in fat-tree networks mice flows are very likely to be blocked by elephant flows in switch queues and suffer from long FCT [29–31]. However, in Thor, mice flows do not compete with elephant flows in electrical switches. Moreover, without elephant flows, traffic is more evenly distributed by ECMP. Thus Thor offers better FCT performance for mice flows. The third reason contributing to the low FCT of mice flows is that Thor employs the low-diameter flattened butterfly topology, which reduces the hop count between servers in the electrical network.

For the elephant flows, Thor achieves 68% to 23% lower average FCT compared to fat-tree when the load varies from 0.1 to 0.6 as shown in Fig. 7(b). The following reasons lead to this improvement. First, optical circuit switching can deliver elephant flows faster than packet switching once an optical connection is established. In an electrical network, every packet experiences routing delay, queuing delay, and O/E/O conversion delay at each hop. Yet in the optical network, a packet only experiences the O/E/O conversion delay at the source and destination servers. Second, fast circuit switching technology ensures that Thor maintains high bandwidth utilization with elephant flows. The use of multiple wavelengths further increases the capacity of the optical network. The average FCT of elephant flows increases faster for loads beyond 0.68 in Thor. This is because the network is mostly saturated at this offered load. When the offered load is larger than 0.68, the increased traffic only raises the collision probability of the optical connections and thus resulting in a lager setup delay and an obvious rising tail in Fig. 7(b). Finally Fig. 7(c) shows the average FCT across all flows. Clearly the overall FCT of Thor is much better than that of fat-tree.

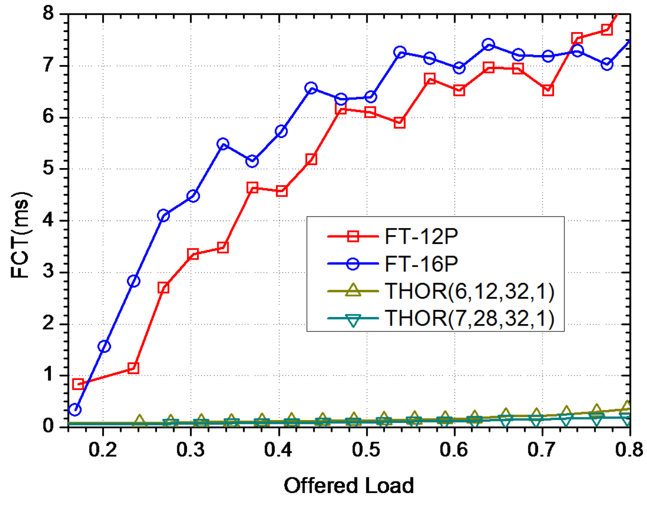
We then evaluate packet delay. As shown in Fig. 7(d), Thor maintains a very low delay of about 2.3 *us* before the offered load of 0.68. After this load, the network becomes saturated and the delay increases significantly from a few microseconds to about 1.2 *ms*. As the compared network, fat tree is saturated at the load of



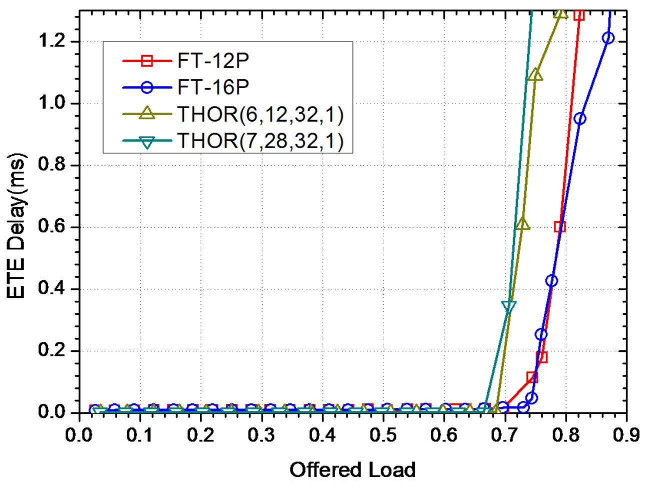
(a) Mice flow FCT



(b) Elephant flow FCT



(c) FCT of all flows



(d) Packet delay

Figure 7: FCT and packet delay performance of Thor and fat-tree.

about 0.75. It does not achieve the full offered load which equals to the bandwidth of server's NIC since the ECMP routing algorithm cannot evenly distribute the workload to all available paths when the traffic is mixed of mice and elephant flows[32]. Note as fat tree is a non-blocking network and the saturation point actually reflects the maximum network capacity, it can be concluded that Thor can deliver the bisection bandwidth that is about 90% of an non-blocking network. This results indicates that Thor outperforms prior optical circuit data center networks. For example, under the all-to-all communication pattern, the bisection bandwidths of OSA[8] and WaveCube[10] are about 58% and 75% of the nonblocking network respectively. Thor achieves this improvement because of the following reasons. First, the low-radix optical switches and the distributed control system enable an optical connection to be established and teared down in tens of microseconds. This greatly improves the bandwidth utilization of the optical links. second, the optical topology provides enough network ports to servers. The elephant flows will not be blocked at the ingress or egress port. Third, Thor provides an efficient way to use the wavelengths. Without increasing too much cost, we can deploy as many wavelengths as possible to reduce the blocking ratio of the optical network.

6 CONCLUSIONS

In this paper, we present Thor, a server level optical/electrical hybrid network for data center. To address the scalability and performance issues of the traditional rack-level optical interconnects, Thor spreads the optical connections to the server-level by employing the low-radix but high scalable topology. Further considering the distinct properties of the electrical packet switching, Thor uses a different topology to form the electronic network. Then at the control plane, Thor uses a distributed control system integrated in the electronic network, which eliminates the processing bottlenecks of the centralized control system and avoids the single point of failure. Finally, Thor develops a new optical switch structure to better exploit the multiple wavelengths. Our evaluations shows that THOR is able to deliver 90% of the non-blocking bandwidth and improve the flow completion time significantly.

Future work: At present, we just make some initial evaluation on the proposed architecture. A deeper and comprehensive evaluation involving more comparison and performance metrics will be made in future. We also plan to build a small-scale test bed to investigate the practical deployment issues. Moreover, the traffic scheduling problems, such as the efficient flow identification and load balance schemes in traffic variable environment[33], needs to be further studied.

7 ACKNOWLEDGEMENTS

The authors greatly thank the anonymous reviewers for their insightful feedback. This work is partially supported by the Research Grants Council of Hong Kong under Project GRF 11202315, ECS 21201714, and CRF C7036-15G, the National Science Foundation of China under Grant No. 61472300, the 111 Project under Grant No. B08038, and the Fundamental Research Funds for Central Universities Grant.

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 63–74, Seattle, WA, USA, 2008. ACM.

[2] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, Barcelona, Spain, 2009. ACM.

[3] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 63–74, Barcelona, Spain, 2009. ACM.

[4] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 75–86, Seattle, WA, USA, 2008. ACM.

[5] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*, 40(4):339–350, 2010.

[6] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. c-through: part-time optics in data centers. In *ACM SIGCOMM*, pages 327–338, New Delhi, India, 2010. ACM.

[7] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. *SIGCOMM Comput. Commun. Rev.*, 43(4):447–458, 2013.

[8] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. Osa: an optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Trans. Netw.*, 22(2):498–511, 2014.

[9] Xiaohui Ye, Yawei Yin, S. J. B. Yoo, Paul Mejia, Roberto Proietti, and Venkatesh Akella. Dos: a scalable optical switch for datacenters. In *the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 1–12, La Jolla, California, 2010. ACM.

[10] Kai Chen, Xitao Wen, Xingyu Ma, Yan Chen, Yong Xia, Chengchen Hu, and Qunfeng Dong. Wavecube: A scalable, fault-tolerant, high-performance optical data center architecture. In *IEEE INFOCOM*, pages 1903–1911, HK, 2015. IEEE.

[11] Yu Gong, Xuezhi Hong, Yang Lu, Sailing He, and Jiajia Chen. Passive optical interconnects at top of the rack: offering high energy efficiency for datacenters. *Optics Express*, 23(6):7957–7970, 2015.

[12] Rodney S. Tucker. Scalability and energy consumption of optical and electronic packet switching. *Journal of Lightwave Technology*, PP(99):1–1, 2011.

[13] Henrique Rodrigues, Richard Strong, Alper Akyurek, and Tajana. S. Rosing. Dynamic optical switching for latency sensitive applications. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 75–86, Oakland, CA, 2015. IEEE.

[14] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: measurements and analysis. In *the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208, Chicago, Illinois, USA, 2009. ACM.

[15] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *the ACM SIGCOMM*, pages 63–74, New Delhi, India, 2010. ACM.

[16] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. In *the 37th annual international symposium on Computer architecture*, pages 338–347, Saint-Malo, France, 2010. ACM.

[17] Sangyoon Han, Tae Joon Seok, N. Quack, Byung wook Yoo, and M. C. Wu. Monolithic 50x50 mems silicon photonic switches with microsecond response time. In *Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, San Francisco, CA, 2014. OSA.

[18] Amin Vahdat, Mohammad Al-Fares, Nathan Farrington, Radhika Niranjan Mysore, George Porter, and Sivasankar Radhakrishnan. Scale-out networking in the data center. *IEEE Micro*, 30(4):29–41, 2010.

[19] A. Wonfor, H. Wang, R. V. Penty, and I. H. White. Large port count high-speed optical switch fabric for use within datacenters. *Journal of Optical Communications and Networking*, 3(8):A32–A39, 2011.

[20] A. R. Curtis, W. Kim, and P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *IEEE INFOCOM*, pages 1629–1637, Shanghai, 2011. IEEE.

[21] Dennis Abts and John Kim. High performance datacenter networks: Architectures, algorithms, and opportunities. *Synthesis Lectures on Computer Architecture*, 6(1):1–115, 2011.

[22] Ching-Tien Ho and S. Lennart Johnsson. Spanning balanced trees in boolean cubes. *SIAM Journal on Scientific and Statistical Computing*, 10(4):607–630, 1989.

[23] Herbert Sullivan and T R Bashkow. A large scale homogeneous fully distributed parallel machine. In *the 4th annual symposium on Computer architecture*, pages 105–117. ACM, 1977.

[24] Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Kieran Mansley. Characterizing 10 gbps network interface energy consumption. Technical report, University of Cambridge, Computer Laboratory, July 2010 2010.

[25] Kostas Christodoulopoulos, Diego Lugones, Kostas Katrinis, Marco Ruffini, and Donal O'Mahony. Performance evaluation of a hybrid optical/electrical interconnect. *IEEE/OSA Journal of Optical Communications and Networking*, 7(3):193–204, 2015.

[26] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A cost comparison of datacenter network architectures. In *the 6th International conference*, pages 1–12, Philadelphia, Pennsylvania, 2010. ACM.

[27] Muhammad Imran, Martin Collier, Pascal Landais, and Kostas Katrinis. Hosa: hybrid optical switch architecture for data center networks. In *the 12th ACM International Conference on Computing Frontiers*, pages 1–8, Ischia, Italy, 2015. ACM.

[28] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280, Melbourne, Australia, 2010. ACM.

[29] Hong Xu and Baochun Li. Repflow: Minimizing flow completion times with replicated flows in data centers. In *IEEE INFOCOM*, pages 1581–1589, Toronto, ON, 2014. IEEE.

[30] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. pfabric: minimal near-optimal datacenter transport. In *the ACM SIGCOMM*, pages 435–446, Hong Kong, China, 2013. ACM.

[31] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. Detail: reducing the flow completion time tail in datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 42(4):139–150, 2012.

[32] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *the 7th USENIX conference on Networked systems design and implementation*, pages 19–19, San Jose, California, 2010. USENIX Association.

[33] Fangming Liu, Jian Guo, Xiaomeng Huang, and John. C. S. Lui. eba: Efficient bandwidth guarantee under traffic variability in datacenters. *IEEE/ACM Transactions on Networking*, 25(1):506–519, 2017.