IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 22, 2019, accepted March 5, 2019, date of publication March 15, 2019, date of current version April 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905354

PETASCALE: A Scalable Buffer-Less All-Optical Network for Cloud Computing Data Center

XIAOSHAN YU[1,2], HUAXI GU[1], KUN WANG[3], AND SHANGQI MA[1]
[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710054, China
[2]Science and Technology on Communication Networks Laboratory, Shijiazhuang 050081, China
[3]Department of Computer Science, Xidian University, Xi'an 710054, China

Corresponding author: Huaxi Gu (hxgu@xidian.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61634004 and Grant 61472300, in part by the Fundamental Research Funds for the Central Universities under Grant JB170107 and Grant JB180309, in part by the China Postdoctoral Science Foundation under Grant 2018M633465, in part by the Key Research and Development Plan of Shaanxi Province under Grant 2017ZDCXL-05-01, and in part by the open found project of Science and Technology on Communication Networks Laboratory under Grant HHS19641X001.

ABSTRACT Optical packet switching is considered as a long-term solution for future data center networks due to their technological strengths, including high throughput, fine switching granularity, and excellent flexibility. However, most prior optical designs are either difficult to scale, costly to solve the packet collision, or inflexible to exploit the multiple wavelengths to increase the end-to-end connectivity. To this end, we present PETASCALE, a scalable, lossless, and high-performance optical data center network. By embedding *full-mesh* into a *bipartite graph*, PETASCALE is able to scale beyond 60 000 servers while achieving up to 2.5 and 2 times saving in a number of switches and cables, respectively, compared with fat tree topology. PETASCALE leverages the negative acknowledgment (NACK) and retransmission scheme to solve the packet collision, thus eliminating the complex and costly buffers. Then by leveraging a state retention mechanism, the contention can be notified over multiple hops in near real time. To further improve the bandwidth utilization, a wavelength routing algorithm is proposed to restrict the packet collision domain within the source pod. Moreover, PETASCALE designs a novel switch structure to support both the multi-hop NACK and wavelength routing. Our simulation results show that PETASCALE is able to reduce the end-to-end latency by more than 50% compared with an electrical fat tree network. For throughput, PETASCALE can deliver about 89% bisection bandwidth of a non-blocking network under uniform traffic pattern. While under the local traffic pattern, it can even achieve higher throughput than fat tree and other optical designs.

INDEX TERMS Data center network, optical packet switching, optical switches, topology.

I. INTRODUCTION

With the rapid development of web service, cloud computing, and big data applications, data center network faces the great challenges of connecting tens of thousands of servers with intensive data interaction. To meet the increasing bandwidth requirement, new data center networks, such as fat tree [1], Portland [2], VL2 [3], DCell [4], BCube [5], and Jupiter [6], have been proposed. Although effectively improving the connectivity, most of above designs adopt multi-layered topology, which may lead to complexity in wiring, routing, and traffic engineering. Moreover, a large amount of switches, transceivers, NICs, and fibers may increase the capital expenditures (CAPEX) and power consumption significantly.

The associate editor coordinating the review of this manuscript and approving it for publication was Ting Wang.

To this end, recent efforts have been proposed by using optical interconnects to flatten the architecture and deliver high bandwidth and throughput in the network. Initially, most of the optical designs, such as Helios [7], c-Through [8], OSA [9], Mordia [10], WaveCube [11], OPMDC [12], OGDCN [13], Multidimensional Switching Structure [14], Archon [15], and RotorNet [16], adopt Optical Circuit Switching (OCS) because this technology matures earlier than other optical switching technologies and has been supported by a large number of commodity devices. However, the slow configuration time ($10 \sim 100$ *ms*) [17] or control overhead for path setup limit the deployment potential of above structures in practice: only a large portion of traffic contains bulky data that lasts enough time to offset the configuration delay, the bandwidth of optical circuits can be utilized effectively. Thus most OCS networks still need

an additional electrical network to deliver the time-sensitive mice flows.

Compared to OCS, Optical Packet Switching (OPS) promises a more flexible operation on almost arbitrary switching granularity [18]–[20] and eliminates the overhead of traffic grooming and path setup. Thus OPS actually provides a long-term solution for the flexible, high bandwidth, and power efficient data center network. Moreover, with the development of silicon photonic integration technology, a variety of photonic elements [21], such as AWGR (Arrayed Waveguide Grating Router), SOA (Semiconductor Optical Amplifier), MZI (Mach-Zehnder Interferometer), Microring resonators, and PLZT (plumb-lanthanum-zirconate-titanate), are able to switch optical signal at packet-level granularity. Thus pure OPS or OPS/OCS hybrid networks have been introduced for data center. The representative designs include DOS [22], Recirculation Buffer Modules [23], OPSquare [24], [25], LIONS [26], H-LIONS [27], [28], Lightness [29], [30], and hybrid broadcast-and-select/wavelength routing architecture [31]. While revealing the potential and significance of building all-optical data center networks, above designs may suffer from the following issues in practice:

A. LOW SCALABILITY OF THE TOPOLOGY

Most prior architectures employ a central OPS switch to connect all servers or racks. The low port count may directly limit the scalability of the network. Although we can naturally expand the number of network ports by interconnecting multiple OPS switches as a Clos topology. Too many switches and fibers are required for this expansion. In addition, it becomes more complex to determine the contention-free paths in Clos topology. OPSquare and H-LIONS enable much better scalability by employing hierarchical topologies. However, they still need the power-consuming OEO (optical-to-electrical-to-optical) conversions in the relay racks.

B. HIGH COST OF THE COLLISION RESOLUTION

Because of a lack of effective buffering schemes, OPS networks face significant challenges in handling the packet collision. Especially data center are now expected to be lossless to provide high throughput and support storage applications [32]. Some buffering structures have been proposed for optical switches. However, the electrical buffer modules suffer from high power consumption and complex memory structures [33], while the optical ones usually require a large number of resources including switching ports, fiber delay lines (FDL), tunable filters, and optical amplifiers, but only provide very limited storage capacity [23], [34], [35]. Moreover, how to effectively eliminate the buffer overflow remains a challenge. An end-to-end scheduling for every packet delivery may be an alternative way to avoid the collision and loss [36], [37]. However, the substantial control overhead may hinder the real implementation of these scheduling mechanisms in large-scale data center.

C. INFLEXIBLE WAVELENGTH ROUTING OVER MULTIPLE HOPS

Wavelength division multiplexing (WDM) carves up the huge bandwidth in a single fiber into non-interfering channels, thus can alleviate the blocking and improve connectivity. However, existing OPS switches cannot support a flexible wavelength routing over multiple hops. The SOA-based switches are wavelength transparent, while the AWG-based structures tightly bound each wavelength to every input-output port pair and thus cannot maintain the wavelength continuity over the whole lightpath.

To overcome above issues, we propose PETASCALE, a scalable, buffer-less, and lossless OPS network for data center. It develops a two-layer hierarchical topology to provide optical connections directly to tens of thousands of servers, thus solving the scalability issue while maintaining a low cost of devices and cables. Then a multi-hop all-optical negative acknowledgment scheme is designed to achieve the lossless delivery of optical packets, without using any electrical or optical buffers. Moreover, to reduce the blocking ratio and increase the bandwidth utilization, we propose a new switch structure and develop the corresponding wavelength routing algorithm.

Overall, this paper makes the following contributions:

- We propose PETASCALE, a scalable, low latency, and flat data center network by employing the multi-hop optical packet switching and two-layer hierarchical topology. More specifically, by embedding the *full-mesh* topology into *complete bipartite graph*, PETASCALE can easily provide all-optical connections to hundreds of thousands of servers, while maintaining low equipment cost and wiring complexity. Moreover, it can deliver high throughput and low latency by exploiting the wavelength routing and buffer-less optical packet switching schemes.
- We develop a contention resolution strategy which eliminates the complex buffering structures by the multi-hop negative acknowledgments (NACK) and retransmission scheme. By exploiting the bidirectional transparency of photonic devices, the forwarding channel (from input to the output port) of an optical switch can also be used to transmit singles backward (from out to input port). Further by leveraging a state retention mechanism, the NACK signal can be counter-propagated over multiple hops. Then to reduce the probability of packet collision, a wavelength routing algorithm is proposed. Moreover, a new optical switch is designed to implement the proposed multi-hop NACK and wavelength routing schemes.
- We evaluate the performance of PETASCALE through extensive simulations. The simulation results reveal that under the uniform traffic pattern, PETASCALE achieves better end-to-end delay and throughput than H-LIONS, and can deliver high bisection bandwidth that is 89% of the non-blocking network. Under the local traffic pattern, PETASCALE can achieve

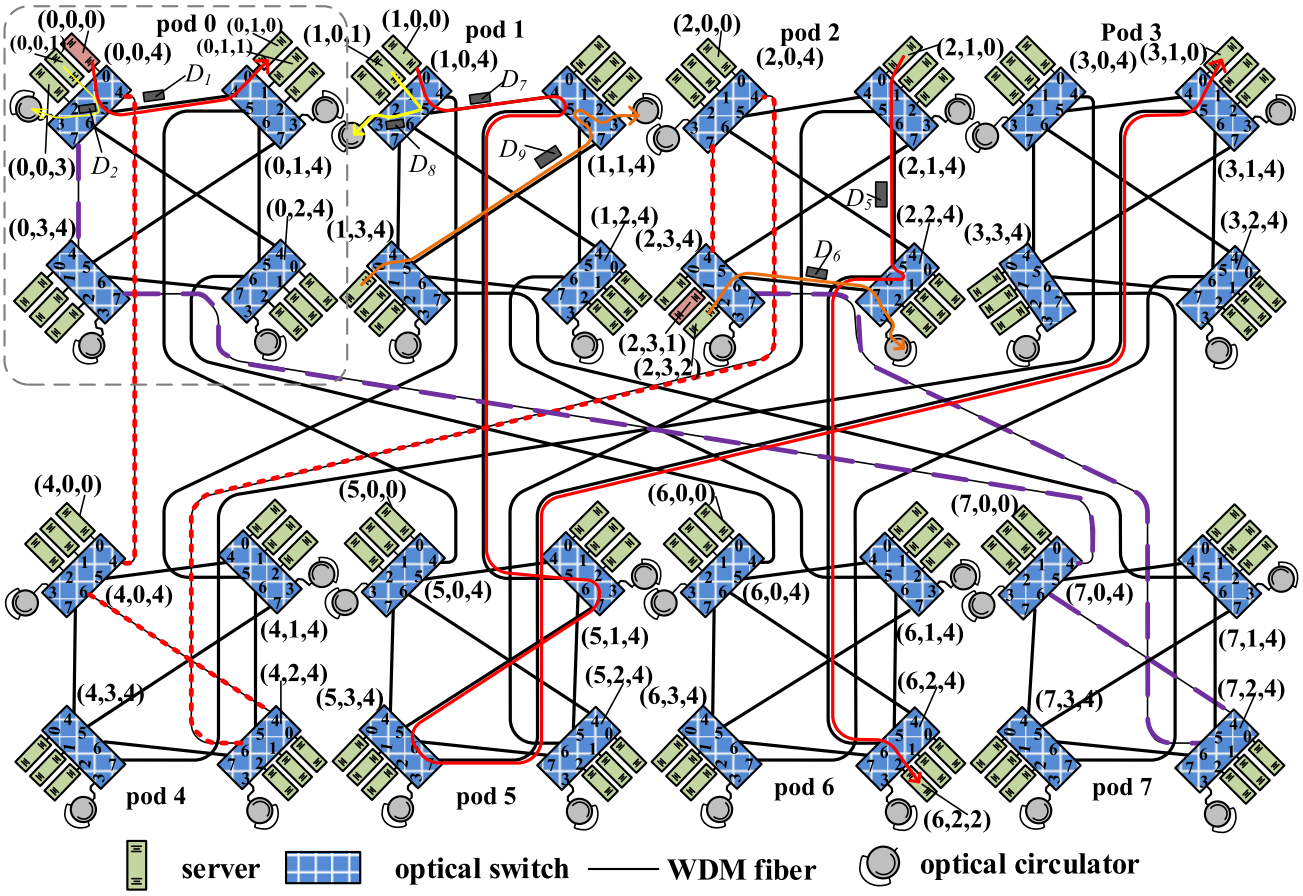


FIGURE 1. The structure of PETASCALE.

salient improvement than prior electrical and optical networks.

The remainder of this paper is organized as follows. Section II introduces the topology and addressing rules of PETASCALE. Section III presents the routing algorithm for the intra- and inter-cluster packets. Section IV describes the strategies to enable a reliable and high-throughput delivery in buffer-less PETASCALE. Section V describes the structure of the new optical switch which is able to support the wavelength switching and multi-hop NACK scheme. Section VI evaluates PETASCALE in terms of end-to-end delay and throughput. Finally, section VII concludes the paper.

II. ARCHITECTURE

A. OVERVIEW

To diminish the latency and management overhead, new data center interconnects should be relatively flat by reducing the network layers and deploying fewer switches. Thus we adopt a two-layer composite graph with only intra- and inter- cluster interconnections to achieve this goal. For the intra-cluster communication, we choose *full-mesh* topology to minimize the hops between local servers. While for the inter-cluster communication, we employ *complete bipartite graph* to save the cost of fibers and switching ports. Although compared to fat tree, hypercube, BCube, and flattened butterfly, *complete bipartite graph* has a smaller bisection width, its bandwidth and connectivity can be significantly enhanced by optical switching and WDM technology. Next we will discuss the detailed structure.

As mentioned above, PETASCALE adopts a cluster-based two-layer hierarchical topology to optically interconnect all servers and k-port switches. As shown in Fig. 1, the network consists of k clusters (called pods) and each cluster contains $k/2$ optical switches. At the bottom layer, each optical switch uses $(k/2-1)$ ports to connect the servers. The remaining $(k/2+1)$ ports can be classified into three types based on their different functions, which include $(k/2-1)$ local ports, one global port and one reflective port. The reflective port is used to solve the packet contention while the other two types of ports, the local and global ports, are used to connect other optical switches. Specifically, each optical switch uses its reflective port to connect an optical circulator. Then it uses its local ports to connect other $(k/2-1)$ switches in the same pod, one port for each switch. Thus in an individual pod, the $k/2$ optical switches uses their local ports to connect each other to form a *full-mesh* topology. The global port of each switch is used to connect an optical switch in another pod. All of the switches in one pod collectively provide $k/2$ global

TABLE 1. Comparison of different data center network topologies.

	PETASCALE	fat tree	FBfly	BCube	DCell	WaveCube	H-LIONS	OPSquare
N_s	$k^3/4-k^2/2$	$k^3/4$	$c\alpha^{\beta-1}$	k^{n+1}	$\Lambda^\dagger$	$c\alpha^\beta$	$W\cdot k^3/8$	$c\cdot k^2$
N_w	$k^2/2$	$5k^2/4$	$\alpha^{\beta-1}$	$k^n(n+1)$	$\Lambda^\dagger/k$	$2\alpha^\beta$	$(W+1)k^2/4+Wk/2$	k^2+2k
Diameter	6	6	$\beta+2$	$n+1$	$<2^{n+1}-1$	$\beta\lfloor\alpha/2\rfloor$	4	4
Bisection Width	$k^2/2$	$k^3/8$	$\alpha^\beta/4$	$k^{n+1}/2$	$\Lambda^\dagger/4\log_k\Lambda^\dagger$	$2\alpha^{\beta-1}$	$Wk^2/8$	$k^2/2$

$*$ N_s is the total number of servers; N_w is the total number of switches; n is the number of levels for BCube and DCell; k is the number of switch ports; FBfly is short for "flattened butterfly", which can be described as α-ary β-dimension structure with c servers per switch; WaveCube is actually an α-ary β-dimensional cube with c servers per rack; H-LIONS exploits k-port passive AWGRs for intra-cluster interconnection and uses W-port active AWGRs to connect $k/2\times W$ clusters; OPSquare is built with k-port switches and each rack contains c servers; $\Lambda^\dagger=(k+1)^{2^n}-1$;

ports to connect the switches in other $k/2$ pods. If the internal links are ignored, each pod can be viewed as a virtual switch with $k/2$ global ports. Then all of the k pods are equally divided into two groups G_{up} and G_{down}, each containing $k/2$ pods. Each pod in one group uses every global port to connect each of the $k/2$ pods in the other group. On the top level, any two pods in the two different groups are connected each other to form a *complete bipartite graph*. Fig. 1 shows an example with 8-port optical switches. Here each switch uses three ports (*i.e.*, port 0, 1, *and* 2) to connect the servers, and uses a reflective port (*i.e.*, port 3) to connect an optical circulator. Then the four switches in the same pod uses their local ports to form an all-to-all interconnection. Further in a higher level, pods 0, 1, 2, and 3 form the group G_{up} and pods 4, 5, 6, and 7 form the group G_{down}. Every pod in G_{up} is connected to each pod in G_{down} through the global links.

B. ADDRESSING AND CONNECTION RULES

Like BCube [5] and DCell [4], PETASCALE employs the customized addressing rules to identify the location of switches and servers. It is feasible because data centers are mainly built and operated by a single entity, and these customized communication strategies can effectively improve the network performance with less control overhead. In PETASCALE each switch is associated with a 3-tuple (*pod_id, switch_id,* $k/2$). The *pod_id*, in $[0, k-1]$, starting from the left to right, top to bottom, is the pod number in the network. The *switch_id*, in $[0, k/2-1]$, arranged clockwise, denotes the position of that switch in the pod. The last element in the 3-tuple, $k/2$, is a constant and used as an identity for the switch. Each server is assigned an address of the form (*pod_id, switch_id, server_id*), where the *pod_id* and *switch_id* are inherited from the directly connected switch, and the *server_id*, in $[0, k/2-2]$, from the top to bottom, is the label identifying the severs in same switch. For an optical switch, its ports are numbered as $0, 1, 2, \ldots, k-1$. The ports $0, 1, 2, \ldots, (k/2-2)$ are used to connect the local servers and port $(k/2-1)$ is used to connect an optical circulator. The rest of the ports are used to connect other switches. The detailed connection rules between two switches $(i, j, k/2)$ and $(u, v, k/2)$ are described as follows:

Rule 1: if $i=u$, then the port $(v+k/2)$ of switch $(i, j, k/2)$ is connected to the port $(j+k/2)$ of switch $(u, v, k/2)$;

Rule 2: if $i<k/2,\ u\geq k/2,$ *and* $i=v$, then the port u of switch (i, j, k) is connected to the port $(i+k/2)$ of the switch $(u, v, k/2)$;

Rule 3: if $i\geq k/2,\ u<k/2,$ *and* $i=v$, then the port $(u+k/2)$ of switch $(i, j, k/2)$ is connected to the port i of switch $(u, v, k/2)$.

In above connection rules, **Rule 1** defines the local connection between two switches in the same pod. For example, in Fig. 1, switches (0, 0, 3) and (0, 2, 4) have the same *pod_id*, then these two switches are connected by a local link, from port 6 of switch (0, 0, 3) to port 4 of switch (0, 2, 4). **Rule 2** and **Rule 3** define the global connection between two switches in the different pods. For example, in Fig. 1, switches (0, 0, 3) and (4, 4, 0) are in groups G_{up} and G_{down} respectively. Then based on **Rule 2**, a global link is connected from the port 4 of switch (0, 0, 3) to the port 4 of switch (4, 4, 0). In addition, if the relations of two switches are not adhered to above three rules, there is no direct connection between these two switches.

C. SCALABILITY ANALYSIS

To implement a more easy control of the optical interconnects, most early designs (*i.e.*, Helios, c-Through, DOS, and OSA) adopt the simple topologies including hub-to-spoke, ring, and spine-leaf. The scalability of these topologies is limited by the port density or wavelength number. Some new optical interconnects, such as WaveCube, H-LIONS, and OPSquare, expand the network capacity by employing more scalable topologies. Meanwhile, pure electrical structures, such as fat tree, flattened butterfly, BCube, and DCell, can be scalable to accommodate hundreds of thousands of servers. Thus we compare PETASCALE with these architectures. Table 1 shows the comparison results. Among these designs, WaveCube and OPSquare actually employ k-ary n-dimensional cube (in OPSquare, the dimensional number is fixed at 2) to scale the network capacity. However, limited by the node degree, these two structures maintain a moderate scalability with up to thousands of network ports and thus can only provide the rack-level optical connections. The recursively defined topologies, such as BCube and DCell, can be expanded to a massive scale with mini-switches. However, the sizes of these two structures expand explosively as the network level increases, and it is hard to control the

number of increased network ports at every expansion. The tree-based and hierarchical topologies, such as fat tree, FBfly, H-LIONS, and PETASCALE, enable a relatively smoother expansion. FBfly usually requires high values of α and β to maintain the favorable network features such as low latency and high throughput, which may result in a strong demand for no-commercial high port-count switches. H-LIONS has a higher scalability than fat tree and PETASCALE. At the same time it uses more switches, transceivers, and fibers to build the network. Fat tree and PETASCALE make better trade-off among many factors including the scalability, connectivity and homogeneity. Further compared to fat tree, PETASCALE achieves better cost-efficiency by using 60% and 50% fewer switches and fibers. Although this benefit is achieved at the expense of relatively lower bisection width, PETASCALE is still able to gain higher bandwidth by using the WDM technology.

III. ROUTING IN PETASCALE

For OPS network, a simple and efficient routing algorithm with low time-complexity is crucial to accelerate the switching speed and reduce the bandwidth waste. Exploiting the regular structure and well-defined connection rules of PETASCALE, a fast routing algorithm is proposed. Each switch makes routing decision based on the source and destination addresses of the incoming packets. The pseudo code of this routing algorithm is shown in Algorithm 1. Specifically, the traffic in PETASCALE can be classified as the *intra-pod*, *intra-group*, and *inter-group* traffic. The packets with the same source and destination *pod_id* are defined as the *intra-pod* traffic. These packets can be directly forwarded from the source switch to the destination switch. The output port at each hop can be calculated by the pseudo codes listed in lines $1-4$ in Algorithm 1. As mentioned in section II-A, all pods in PETASCALE are divided into two groups. For a packet, if the source and destination *pod_id*s are in the same group, it belongs to the *intra-group* traffic. Otherwise, if the source and destination *pod_id*s are in two different groups, this packet belongs to the *inter-group* traffic. It mainly involves two steps to route an *inter-group* packet. First, this packet is routed in the source pod, from the source switch to the intermediate switch which has a global link connected to the destination pod. The pseudo codes in lines $08-12$ describe this process. Then the *inter-group* packet is routed in the destination pod and its output is calculated based on the pseudo codes in lines $01-04$. For the *intra-group* packets, there are actually two disjoint routing paths which can be called "*pod-first-path*" and "*switch-first-path*" respectively. For the *pod-first-path*, the *intra-group* packet is firstly routed to the intermediate pod the source switch directly connected to. Then it is forwarded to the destination pod and further to the destination switch. While for the *switch-first-path*, the *intra-group* packet is firstly routed to the switch which is in the source pod but has the same *switch_id* as the destination server. Then it is forwarded to the intermediate pod this switch directly connected to. Finally through the global link between the intermediate pod and the destination pod, this packet directly arrives at the destination switch. An example shown in Fig.1 illustrates these two paths from the source server $(0, 0, 0)$ to the destination server $(2, 3, 1)$. The *pod-first-path* is routed through switches $(0, 0, 4)$, $(4, 0, 4)$, $(4, 2, 4)$, $(2, 0, 4)$, and $(2, 3, 4)$, while the *switch-first-path* is routed through switches $(0, 0, 4)$, $(0, 3, 4)$, $(7, 0, 4)$, $(7, 2, 4)$, and $(2, 3, 4)$. Both of these two paths are the shortest paths between the source and destination servers. However, if two servers in the same pod simultaneously send packets to the same destination server, the collision will occur at the different switches based on these two routing paths. For example, in Fig. 1, when two servers $(0, 1, 0)$ and $(0, 2, 1)$ simultaneously send packets to the server $(2, 3, 1)$, these two packets will collide at switch $(2, 3, 4)$ if traveling along the *pod-first-path*. Otherwise, they will collide at switch $(0, 3, 4)$ if adopting the *switch-first-path*. Since our contention resolution strategy which will be discussed in the next section can solve packet collision with less cost when they are more closer to the source server, the *switch-first-path* is adopted by PETASCALE. Accordingly, the pseudo code in line 08 describes routing the *intra-group* packets within the source pod. The codes in lines $10-13$ and in lines $01-02$ describe routing the *inter-group* packets in the intermediate pod and destination pod respectively. Note based on this routing algorithm, any incoming packet can acquire its output port after performing the logical comparison twice, regardless of the network sizes and traffic types. This constant time-complexity ensures a fast routing process.

Algorithm 1 Routing a Packet in PETASCALE

Input: source address of a packet: (x_s, y_s, z_s); destination address of the packet: (x_d, y_d, z_d); address of current switch: (x_c, y_c, z_c);
Output: output port: P_{out}

```
1:  if x_c = x_d then                          ▷ routing packet in the destination pod
2:      if y_c = y_d then P_out = z_d
3:      else P_out = y_d + k/2
4:      end if
5:  else if x_c = x_s then                     ▷ routing packet in the source pod
6:      if x_c < k/2 and x_d ≥ k/2 then P_out = x_d
7:      else if x_c ≥ k/2 and x_d < k/2 then P_out = x_d + k/2
8:      elseP_out = y_d + k/2
9:      end if
10: else                                       ▷ routing pacekt in the intermediate pod
11:     if x_d < k/2 then P_out = x_d + k/2
12:     elseP_out = x_d
13:     end if
14: end if
```

IV. THE CONTENTION RESOLUTION STRATEGY FOR OPS

Different from the Optical Circuit Switching (OCS) and Optical Burst Switching (OBS) which use the control packets to make two-way or one-way path reservation before data transmission, OPS tightly couples the control information with the

payload and sends them without path reservation. An optical channel adapter is equipped at each server, taking charge of the packet format translation and time synchronization. Through this adapter, the message is converted into a fixed length optical payload with a label identifying the source and destination addresses. Then at each hop, the optical label is separated by the label extractor, sent to the control module, and then processed in the electrical domain. The payload is temporarily buffered in the FDL, waiting for the routing calculation, contention resolution, and switch allocation. Then the payload is directly forwarded through the crossbar. At the output port, the optical label is reassembled with the payload and transmitted to the next hop. Without path reservation or offset time control, OPS can achieve the similar switching granularity and traffic additivity as the Electrical Packet Switching (EPS). However, how to solve the packet collision is a big challenge for OPS network. Since some storage applications cannot tolerate the packet loss, cloud computing data center now tends to guarantee a reliable end-to-end transmission with no packet dropping. Thus it is impractical to simply drop the collided packets. Intuitively, the buffers can provide an effective way to avoid packet loss. However, there are some limitations in deploying buffers to optical packet switches. The electrical buffer structures require fair amount of memory and transceivers [26] while the optical ones suffer from low storage capacity and inflexible access to the buffered packets. Additionally, the flow control mechanism is needed to prevent the buffer overflow and this may further increase the system complexity and control overhead. The all-optical negative acknowledgment (AO-NACK) scheme [38] provides a promising approach to solve the packet collision in all-optical domain. However, currently this scheme can only be applied to the centralized optical interconnects with the source and destination nodes directly connected to an AWGR-based switching fabric. To solve the packet collision in a more scalable network, the AO-NACK scheme should be extended from a single-hop to multiple hops. PETASCALE realizes this extension by combining the negative notifying technique with a state retention mechanism. Next we detail the design of the multi-hop AO-NACK scheme.

Since most optical switching elements, such as AWGR, SOA, and microring resonator, are bidirectionally transparent, a connection from the input to the output potentially allows the feedback signal to be transmitted in the anti-direction (*i.e.*, from the output to input) [26]. The AO-NACK scheme essentially utilizes this property to build a real-time feedback system. If two packets contend for the same output port, the control module forwards one of these two packets to the required output port while forwards the other packet to the reflective port. The one forwarded to the reflective port now acts as the NACK packet. Through the optical circulator which is equipped at the output of the reflective port, this NACK packet is sent back to the original input port. In a centralized optical network, this NACK packet can easily return back because it just takes a very short period of time (about tens of nanoseconds) to implement the counter-propagating. During this time, the crossbar inherently keeps the connection between the input and reflective port. However, in PETASCALE above operation cannot ensure the NACK packet returns back since the contention may occur at the switch that is multiple hops away from the source server. To make this NACK packet travel back over multiple optical switches, the forwarding path, from the output port of the source server to the switch the collided packet currently arriving at, should be kept unchanged. A state retention mechanism is proposed to achieve this goal. Specifically, when a source server sends a packet, it will estimate the collision zone of this packet. Here the collision zone is used to describe the set of switches at which the packet will probably be blocked by other packets. Based on this estimation, the server will know when the NACK may return back. While the packet is routing in the collision zone, a new sending packet from the same source server may change the switching state of the first-hop switch. Thus to hold the forwarding path, the source server will lock its output port until the prior packet travels out of the collision zone. Similarly, the optical switches along the forwarding path also needs to retain their switching state to ensure the NACK packet can travel back to the source. A state retention mechanism is deployed to hold this bidirectional routing path. When the control module of a switch receives the head of the packet, it calculates the output port and then checks whether the required port has been locked. If yes, the corresponding payload is switched to the reflective port and then transmitted back to the source. Otherwise, the control module further checks whether there are other packets requiring for the same output port. If yes, the control module allocates the output to one of these packets and switches others to the reflective port. If there is no competition, the packet acquires grant for the output port. Finally, the control module calculates the collision zones for the packets which get the required output ports. Then it calculates the duration time to lock the corresponding ports (this time is called locking delay), marks the state of these output ports, and then sends packets to the next hop.

In fact, if there is no WDM technique, the packet will probably collide with other packets at each hop along the path from current arriving switch to the destination server. Since the packets do not experience any queuing delay in PETASCALE, it is easy to calculate the locking delay at each hop. For example, in PETASCALE with $k = 8$ (shown in Fig. 1), a packet is sent from $(0, 0, 0)$ to $(4, 2, 0)$. Based on the routing algorithm, the source server knows there are 4 hops to reach the destination and the packet may be blocked at switches $(0, 0, 4)$, $(4, 0, 4)$, and $(4, 2, 4)$. As shown in Fig. 2, if this packet is sent at time t_0 and then blocked at the first-hop switch $(0, 0, 4)$, this packet will return to the source server at time $t_0 + 2t_r + t_s$, where t_r is the propagation delay and t_s is the latency required for processing the packet head and configuring the crossbar. Similarly, if this packet is blocked at the second-hop switch $(4, 0, 4)$ or the last-hop switch $(4, 2, 4)$, it will return back at time $t_0 + 4t_r + 2t_s$ or

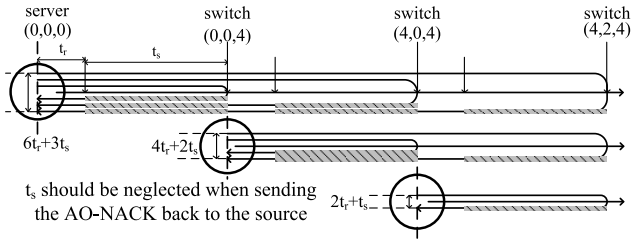


FIGURE 2. The locking delay for each node when sending a packet from (0, 0, 1) to (4, 2, 1). The variable t_s is the time taken for the OPS switch to process the packet head and configure the crossbar, while t_r is the time taken for the head of the optical packet to travel on the fiber.

$t_0 + 6t_r + 3t_s$. To ensure the NACK packet can always be received, the locking delay of the source server is set to $6t_r + 3t_s$. When this packet arrives at switch (0, 0, 4) and successfully acquires the output port, the switch (0, 0, 4) needs to keep the connection state by locking its output port. As there are 3 hops to reach the destination, switch (0, 0, 4) sets the locking delay to $4t_r + 3t_s$. Similarly, the switch (4, 0, 4) sets the locking delay to $2t_r + t_s$ and sends it to the next hop. The last switch does not lock its output port since it either sends the packet to the destination or forwards it back to the source server immediately. Here the locking delay is the longest time required for the NACK packet traveling back to current node. Thus the locked port can be released earlier than expected once the node detects the returned NACK packet.

Although above state retention mechanism enables a reliable delivery over the multi-hop buffer-less network, the link utilization is inefficient because the switches spend more time on waiting for the counter propagating NACK packet rather than forwarding the effective data. Thus we need to shorten the locking delay by reducing the collision probability. WDM can be employed to restrict the collision to a few switches along the routing path. A wavelength assignment algorithm is proposed to meet the following two requirements: first, since the local traffic has been observed as the dominated workload in cloud computing data center [39], the collision zone of the *intra-pod* traffic should be confined to the first-hop switch. Second, for the *intra-group* and *inter-group* traffic, the locking delay should be significantly reduced by restricting their collision zones within the source pod. As shown in Algorithm 2, a specific wavelength is statically assigned to each packet based on its source and destination addresses. Most packet collisions can be avoided by using the different wavelengths. However, as the number of available wavelengths is limited, some packets have to use the same wavelength and certain collisions still exist in the network. Fortunately, the collision zones of these packets have been restricted to the first or second hop from the source server. We can use an example to illustrate this improvement. Based on Algorithm 2, the wavelength assignment for PETASCALE with $k = 8$ is shown in Fig. 3. The allocated wavelengths can be filled in a $k \times k$ compound matrix M_k with the source and destination *pod_id*s as the row and column indexes respectively. The element in the i^{th} row and j^{th} column ($i \neq j$) determines the wavelength used for sending a packet from pod i to pod j. The element of the main diagonal of M_k is a $k/2 \times k/2$ sub-matrix $U_{k/2}$, which further uses the source and destination *switch_id*s as the row and column indexes. The element in the p^{th} row and q^{th} column of the sub-matrix $U_{k/2}$ determines the wavelength used for sending an *intra-pod* packet from switch p to q. This wavelength assignment algorithm effectively alleviates the collision probability in the following aspects:

First, the *intra-pod*, *intra-group*, and *inter-group* traffic will not block each other. As shown in Fig. 3, the *intra-pod* traffic uses the wavelengths $\{\lambda_0, \lambda_1, \lambda_2, \lambda_3\}$. The *intra-group* and *inter-group* traffic uses the wavelengths $\{\lambda_4, \lambda_5, \ldots, \lambda_9\}$ and $\{\lambda_{10}, \lambda_{11}, \lambda_{12}, \lambda_{13}\}$ respectively. Three logical networks are built independently for the *intra-pod*, *intra-group*, *inter-group* traffic, thus a packet can only be blocked by another one with the same traffic type.

Then for the *intra-pod* communication, only two packets with the same source and destination *switch_id*s will probably collide at the first-hop switch. As shown in the sub-matrix $U_{k/2}$ in Fig. 3, the *intra-pod* packets will always use the different wavelengths except for the ones having the same stride length $\sigma_{str}(\sigma_{str} = 0, 1, \ldots, k/2 - 1)$ from the source *switch_id* y_s to the destination *switch_id* y_d. Here the stride length can be calculated by $\sigma_{str} = (y_d - y_s + k/2) mod (k/2)$. If two packets are sent from the different source switches and have the same stride length, they will be routed on the different links. Thus there is no collision between these packets even they use the same wavelength. Only with the same source *switch_id* and stride length, the packets will target for the same destination switch and then collide at the output port of the source switch. For example, in Fig. 1, the packet D_1 from (0, 0, 0) to (0, 1, 0) will collide with packet D_2 from (0, 0, 1) to (0, 1, 1) at the output port 5 of the source switch (0, 0, 4).

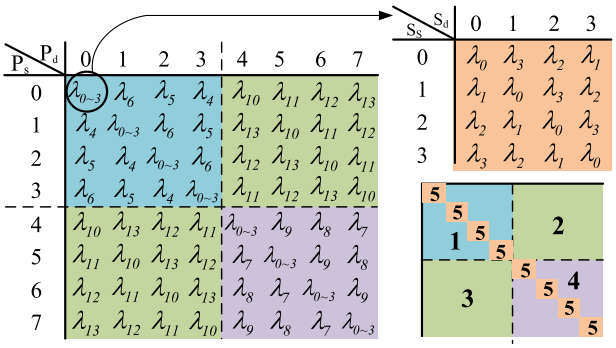


FIGURE 3. The wavelength assignment for PETASCALE with $k = 8$ (P_S is the source *pod_id*, P_d is the destination *pod_id*, S_s is the source *switch_id*, S_d is the destination *switch_id*).

Similarly, for the *inter-group* and *intra-group* communication, the collisions are limited to a fraction of specific packets in the source pod. As shown in the matrix M_k in Fig. 3, the *intra-* and *inter- group* packets use the same wavelength

Algorithm 2 Wavelength Assignment in PETASCALE

Input: the source address of the packet: (x_s, y_s, z_s); the destination address of the packet: (x_d, y_d, z_d);

Output: the index the wavelength channel: w ▷ packet is sent using λ_w

1: **if** $x_s = x_d$ **then** ▷ intra-pod forwarding
2: **return** $w = (k/2 - y_d + y_s) mod (k/2)$
3: **else if** $x_s < k/2$ *and* $x_d < k/2$ **then** ▷ intra-group (in G_{up}) forwarding
4: **return** $w = (k - x_d + x_s) mod (k/2) + k/2 - 1$
5: **else if** $x_s \geq k/2$ *and* $x_d \geq k/2$ **then** ▷ intra-group (in G_{down}) forwarding
6: **return** $w = (k - x_d + x_s) mod (k/2) + k - 2$
7: **else if** $x_s \geq k/2$ *and* $x_d < k/2$ **then** ▷ inter-group forwarding
8: **return** $w = (k - x_d + x_s) mod (k/2) + (3k/2 - 2)$
9: **elsereturn** $w = (k - x_s + x_d) mod (k/2) + (3k/2 - 2)$
10: **end if**

only when they have the same stride length $\sigma_{str} (\sigma_{str} = 0, 1, \ldots, (k/2-1))$ from the source pod x_s to the destination pod x_d, that is $\sigma_{str} = (x_d - x_s + k) mod (k/2)$. If two *inter-* or *intra- group* packets are sent from the different source pods and carried by the same wavelength, these packets are routed on the different links and there is no collision between them. For example, in Fig. 1, packet D_3 is sent from (0, 0, 0) to (1, 0, 0). Meanwhile, packet D_4 is sent from (3, 0, 0) to (1, 0, 0). Although they are carried by the same wavelength λ_6 and relayed by the same intermediate pod, these two packets are routed on the edge-disjoint paths. However, if two *inter-group* packets are sent from the same source pod and carried by the same wavelength, these two packets will firstly be forwarded to the same relay switch in the source pod and then collide at the output port. For example, in Fig. 1, packets D_5 and D_6, from (2, 1, 0) to (6, 2, 2) and from (2, 3, 2) to (6, 0, 1), will collide at the switch (2, 2, 4). For the *intra-group* communication, two packets, with not only the same source and destination *pod_id*s but also the same source and destination *switch_id*s, will collide at the relay switch in the source pod. For example, in Fig. 1 packet D_7 is sent from (1, 0, 0) to (3, 1, 0). Simultaneously, packets D_8 and D_9 are sent from (1, 0, 1) to (3, 1, 1) and from (1, 3, 0) to (3, 1, 2) respectively. Then packets D_7 and D_8 will collide at switch (1, 0, 4). If D_7 wins the competition, it will further collide with D_9 at switch (1, 1, 4). However, once packet D_7 is forwarded out of the source pod, it will never be blocked by other packets along the remaining path.

In summary, exploiting the wavelength routing algorithm, a packet will only be blocked by other ones with the specific source and destination addresses. Since the network does not buffer any packets, the possibility of a packet encountering its competitors can further be confined to the special switches. Specifically, an *intra-pod* packet can only be blocked by other ones with the same source and destination *switch_id*s at the first-hop switch. An *inter-group* packet can only be blocked by other ones with the same source and destination *pod_id*s at the first- or second-hop switch. An *intra-group* packet can only be blocked by the ones sent from the same pod and having the same destination *pod_id* and *switch_id*. This collision occurs at the first or second hop from the source server. Thus exploiting the WDM communication, the source server and corresponding switches need to lock an specific output wave channel rather than the output port. Moreover, this locking state only needs to be maintained for a little while until the corresponding packet is forwarded out of the first or second hop. The detailed state retention algorithm is shown in Algorithm 3.

Algorithm 3 The WDM-Based State Retention Algorithm

Input: the source address of the packet: (x_s, y_s, z_s); the destination address of the packet: (x_d, y_d, z_d); address of current server or switch: (x_c, y_c, z_c); the wavelength this packet using: w; sub-function used to distinguish the type of a packet (intra-pod, intra-group, or inter-group): $T(x, y)$ (x, y are the source and destination *pod_id* respectively)

Output: t^w_{lock}: the time duration to lock the wavelength channel w at output port after sending a packet

1: **if** $x_s = x_d$ **then return** $t^w_{lock} = 0$
2: **else if** $T(x_s, x_d)$ = *intra-group* and $y_s \neq y_d$ **or** $T(x_s, x_d)$=*inter-group* and $y_s \neq x_d mod (k/2)$ **then**
3: **if** $z_c = z_s$ **then return** $t^w_{lock} = 4t_r + 2t_s$ ▷ t_r is the propagation delay while t_s is the routing delay
4: **else if** $x_c = x_s$ and $y_c = y_s$ **then return** $t^w_{lock} = 2t_r + t_s$
5: **else return** $t^w_{lock} = 0$
6: **end if**
7: **else return** $t^w_{lock} = 0$
8: **end if**

V. THE MULTI-WAVELENGTH SWITCH STRUCTURE

Exploiting the multi-hop NACK and wavelength routing, PETASCALE enables an efficient delivery which significantly cuts down the blocking ratio. However, existing OPS switches cannot well support the wavelength-based multi-hop NACK scheme. Specifically, the SOA-based switches are transparent to the wavelengths. While for the AWGR structures, the used wavelength is determined by the input and output ports [22]. To correctly arrive at the destination, a packet needs to change its wavelengths hop by hop. Thus it is hard to assign a specific wavelength to a light path in AWGR based networks. To solve above problem, we design a new switch structure to support the flexible wavelength routing and multi-hop NACK.

The detailed structure of this switch is shown in Fig. 4. Here $(k/2 - 1)$ of the k ports, labeled as $P_0, P_1, \ldots, P_{k/2-2}$, are connected to the servers. One port labeled as $P_{k/2-1}$ is used as the reflective port. The remaining $k/2$ ports, labeled as $P_{k/2}, P_{k/2+1}, \ldots, P_{k-1}$, are connected to other switches. The main switching fabric consists of the bidirectional input and output ports, optical crossbar and control module.

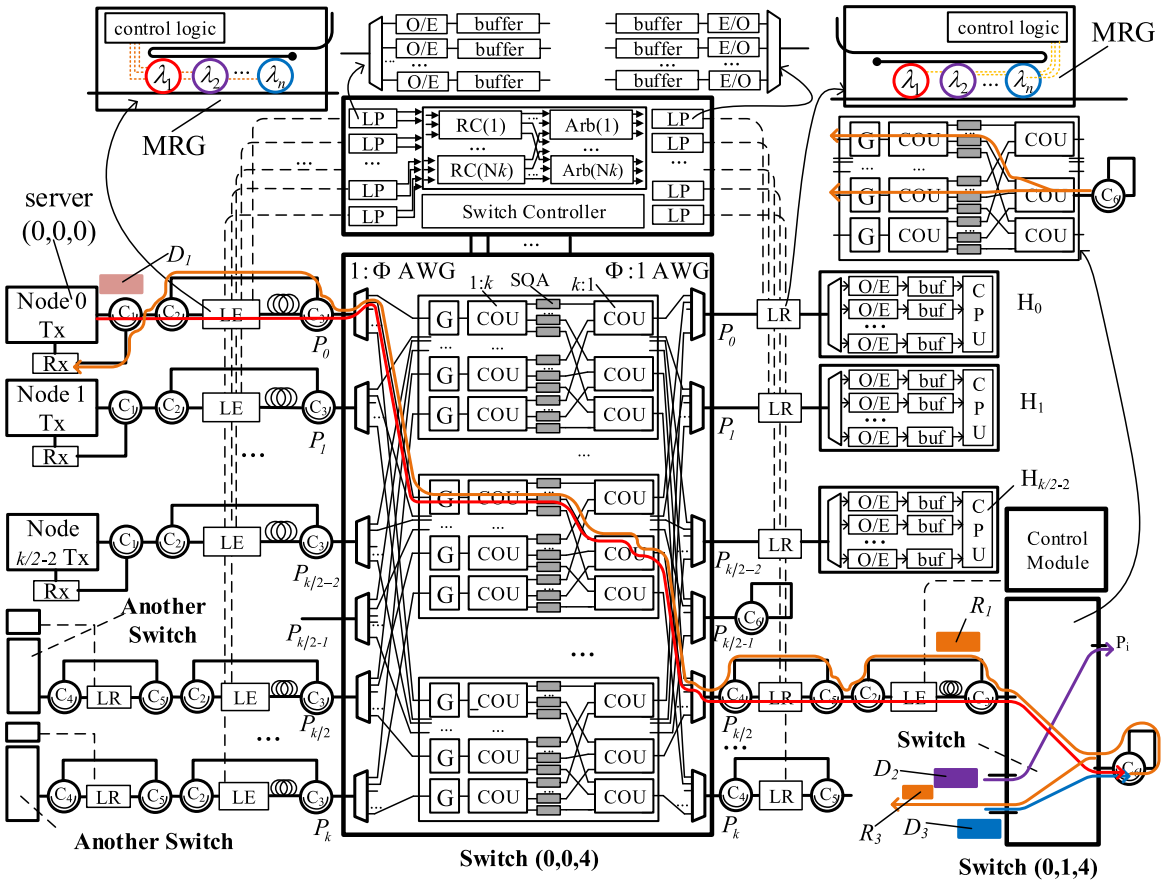


FIGURE 4. The structure of the multi-wavelength optical switch supporting multi-hop AO-NACK.

At the core of the switch are a bidirectional WDM crossbar which is organized in k $1:\Phi$ AWGs ($\Phi = 2k-2$), then Φ $k \times k$ sub-crossbars, and then followed by k $\Phi:1$ AWGs. At the input each $1:\Phi$ AWG is connected to Φ sub-crossbars, used to separate the WDM signal into single wavelength and then deliver each one to the corresponding sub-crossbar. Each sub-crossbar is responsible for switching one wavelength from the input to output. It adopts the SOAs to form a broadcast-and-select structure. Specifically, at the input of the sub-crossbar, an optical amplifier is deployed to compensate the transmission loss. Then the optical signal is passively split by the $1:k$ coupler and gated by SOAs. Each input is logically connected to each of the output by k SOAs and then the links targeting for the same output are assembled by a $k:1$ coupler. After switched by the sub-crossbars, the wavelengths, coming from different inputs but targeting for the same output, are multiplexed by the $\Phi:1$ AWG again.

Although all modules in the crossbar are bidirectionally transparent, some components at the input and output ports are unidirectional. Thus to enable a bi-directional transmission along the whole path, the optical circulators are deployed on the both sides of the unidirectional components, forming short bridges to transmit the backward signal. For example, in Fig. 4 a pair of optical circulators C_2 and C_3 create a bridge over the label extractor (LE) and fiber delay lines (FDL). The optical circulator pair C_4 and C_5 form another bridge over the label re-insertion (LR) module. Here LE and LR are used to extract the packet head at the input and re-install it at the output. Besides used for building the bidirectional links, optical circulators are also used for other purposes: the optical circulator C_6 is used to forward all packets arriving at the reflective port back to their inputs. Optical circulator C_1, equipped at the output of the server, is used to separate the counter-propagating NACK signal from the data packets.

The control module is responsible for configuring the forwarding path between the input and output ports. In each control loop, the label processor (LP) receives the packet heads, makes O/E conversions and then stores these heads. The routing computation (RC) module calculates the output ports based on the head information. Then the packet heads requiring for the same output port and using the same wavelength are sent to the corresponding arbiter for contention resolution. If the arbiter is unlocked, it gives the grant to a randomly selected packet head and rejects others. Then it calculates the locking delay and changes the state based on the calculation result. If the arbiter is locked, it directly rejects all packet heads. All decisions will be sent to the switch controller after finishing switching arbitration. Based on these information, the switch controller configures the SOAs of corresponding sub-crossbars. The granted packets

are forwarded to the required output ports while the rejected packets are forwarded to the reflective port.

The whole switching fabric is bidirectionally transparent. Moreover, the SOA will maintain its ON state for a given time. These two measures enable the NACK packet to be transmitted over multiple hops. For example, in Fig. 4, server (0, 0, 0) sends an *inter-group* packet D_1 to server (5, 0, 0). This packet needs to travel two hops before leaving the source pod. Thus after sending this packet, the server locks its corresponding wave channel and holds the locking state for the duration of $4t_r + 2t_s$. Assume there is no competition at switch (0, 0, 4). The control module of switch (0, 0, 4) sends packet D_1 to switch (0, 1, 4). At the same time, it sets the lock time for the corresponding wave channel of the output port to be $2t_r + t_s$. At the switch (0, 1, 4), three packets D_1, D_2 and D_3, require the same output port and the same wave channel. The control module gives its grant to packet D_2 and switches the packets D_1 and D_3 to the reflective port. As packet D_1 and D_3 use the same wavelength, these two data signals are mixed together at the reflective port and some information is lost. However, since the NACK packet is used to notify the failure of the transmission. An optical pulse which can be detected by the edge detector can deliver this message. Moreover, as the arrival time of the NACK packet can be exactly estimated in a buffleless network. The source server has recorded the sending time of every packet which may conflict with other packets on the path. Based on the wavelength and the return time, the source can easily find which packet the NACK packet refers to. So based on this time-stamp scheme, the merged signal is still able to act as the NACK notification. As shown in Fig. 4, at switch (0, 1, 4) the NACK packet is re-injected into the output of the reflective port and divided into k ways by the optical coupler. Selected by two ON-state SOA gates, one NACK packet named R_1 is switched to the input port of the packet D_1 while the other NACK packet named R_3 is switched to the input port of the packet D_3. As the circulators C_2 and C_3 form a bridge across over the LE and FDL, the NACK packet R_1 only experiences a propagation delay t_r before it arrives at the switch (0, 0, 4). At this moment, switch (0, 0, 4) still keeps the switching state for forwarding data packet D_1, thus R_1 can directly travels from the output port to the input port. Finally the detector equipped at the output port of server detects the NACK packet and triggers the retransmission.

VI. EVALUATION

To study the network performance of PETASCALE, we develop an extensive packet-level simulation based on OPNET simulator. In this section, we first introduce the evaluation methodology, and then present the simulation results.

A. EVALUATION METHODOLOGY

1) TOPOLOGY

Our simulation is mainly based on two network sizes of PETASCALE. The first one, *PS-12P*, totally hosts 360 servers using 12-port optical switches. The second one, *PS-16P*, hosts 896 servers using 16-port optical switches. The non-blocking electrical network fat tree and high scalable optical network H-LIONS [27] are selected as the comparison baselines. For H-LIONS, it exploits passive AWGR all-to-all interconnection for intra-cluster communication and then leverages a flat AWGR topology for inter-cluster communication. In the simulation, H-LIONS has 18 clusters. Each cluster contains 6 AWGR switches, with one acting as a relay node and each of the other AWGRs connecting 6 servers. Thus this topology totally hosts 540 servers.

For fat tree, we assume the link bandwidth is 10 *Gbps* and optical transceivers are equipped for the links between aggregation and core layers. Both PETASCALE and H-LIONS can leverage multiple transceivers to increase the network capacity. For example, in H-LIONS, each server is equipped with $(N-1)$ transceivers originally, where N is the port count of the passive AWGR switch. However, to make a fair comparison, in H-LIONS and PETASCALE, we assume each server is equipped with a single transceiver with tunable wavelengths that carries $10G$ bandwidth.

2) TRAFFIC PATTERNS

In the simulation, the packet generation process follows a Poisson distribution with varying mean arrival rates λ. Then the strength of the offered traffic is described by the product of the arrival rate λ and the average packet transmission time. For each packet, its destination server is generated based on two traffic patterns, that is *uniform pattern* and *local pattern*. In *uniform pattern*, a server will send packets to any other server in the network with uniform probability. While in local pattern, a server will send packets to another server in its cluster or pod with much higher probability (70%), while to anyone else with lower probability (30%).

3) EVALUATION METRICS

We select the packets average end-to-end delay and network throughput as the evaluation metrics. Here the average end-to-end delay refers to the average time taken by a packet to be delivered from the source to the destination server, while the network throughput refers to the average data rate at which packets are successfully received by the destination server.

B. SIMULATION RESULTS

We now discuss the simulation results. Firstly, the end-to-end delay and throughput of different networks versus the offered load under uniform traffic pattern is tested and compared in Fig. 5. It can be observed that the OPS networks, *i.e.*, H-LIONS and PETASCALE, are able to achieve much lower end-to-end delay than electrical networks under the light and moderate traffic load. As shown in Fig. 5 (a), fat tree keeps the average end-to-end delay of about $2.51us \sim 2.86us$ before it is saturated by the heavy traffic load. While H-LIONS and PETASCALE keep the average end-to-end delays of about $0.8483us$ and $1.0664us$ respectively before their saturation loads. Compared to fat tree, H-LIONS and PETASCALE can

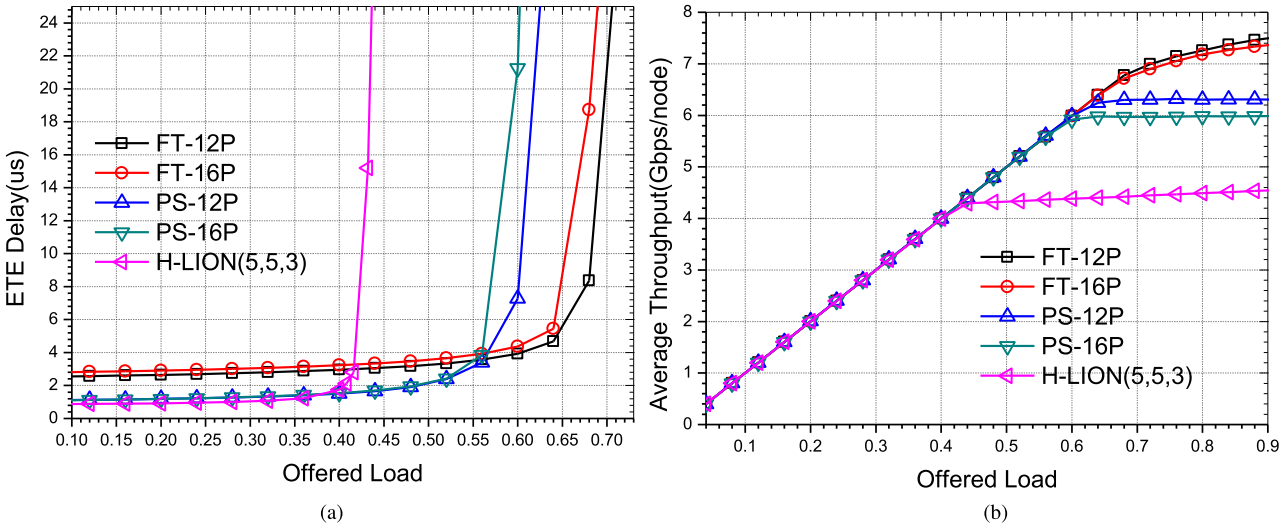


FIGURE 5. The average end-to-end delay (a) and throughput (b) under uniform traffic pattern.

reduce over 50% of the unsaturated end-to-end delay. This is mainly because of the following two reasons. First, packets in OPS networks will always keep propagating once they leave the source. They rarely experience the optical-to-electrical-to-optical (OEO) conversions and store-and-forward operations at the intermediate switches. Thus the OPS switches can actually provide a faster cross connection between the input and output ports. Second, to reduce the cost and eliminate electronic bottlenecks, OPS networks tend to use in-network buffers as few as possible. For example, H-LIONS only deploys electrical buffers in the relay racks. While PETASCALE avoids using any buffer in the switches. Thus the queuing delay, which is a significant contributor to in-network latency in data center, can be minimized by the optical packet switching. For the two optical networks, H-LIONS maintains lower unsaturated end-to-end delay than PETASCALE because it has a shorter network diameter.

For the saturation point, as shown in Fig. 5 (a), H-LIONS is able to keep a low end-to-end delay when the offered load is smaller than 0.42. At this stage, the network has full capacity to deliver the injected traffic. Thus as shown in Fig. 5 (b), the throughput of H-LIONS increases linearly with the increasing offered load. However, when the offered load is larger than 0.42, H-LIONS becomes saturated. The supernumerary packets are accumulated rapidly in the servers and relay racks. And this leads to a sharp increasing of the end-to-end delay of H-LIONS from 0.8483*us* to dozens of microseconds. Similarly, PETASCALE networks, including PS-12P and PS-16P, are saturated at about 0.58. While fat tree networks, including FT-12P and FT-16P, have the highest saturation point of about 0.65. This is not surprising as fat tree is a nonblocking network and its full-bandwidth connectivity can be fully utilized by the uniform traffic. H-LIONS has the lowest saturation point. This is mainly because in H-LIONS the server also participates in forwarding packets to the correct destination besides acts as the traffic generator and sink. For one server, if there are multiple transceivers, the local and relay traffic can be sent simultaneously. However, as in the simulation each server has only a single transceiver, the local and relay traffic will collide frequently for the output of the relay server. PETASCALE performs better than H-LIONS but not as good as fat tree. This is because PETASCALE adopts a different way (i.e. multi-hop NACK) to solve the packet collision, which ensures a more scalable, fast, and transparent network. As a comprise, PETASCALE needs to consume a portion of bandwidth to delivery the notification for packet collision. We also notice that PETASCALE is able to deliver traffic load that is about 89.23% of an non-blocking network. This result outperforms most prior optical data center networks. For example, OSA, WaveCube and Mordia are able to deliver about 58%, 75%, and 87.9% bisection bandwidth of the non-blocking electrical network respectively. This improvement is achieved by the following reasons: first, compared to the optical circuit switching, which is adopted by OSA, WaveCube, and Mordia, the optical packet switching eliminates the overhead of path setup and enables a better bandwidth utilization at almost arbitrary transmission granularity. Second, the wavelength routing algorithm effectively limits the packet collision within the source pod, which means once a packet travels out of the first two hops, it will never be blocked by other packets. Third, compared to AWGR based designs, such as H-LIONS, PETASCALE can exploit more wavelengths by using its SOA based switch structure. For example, when holding 360 servers, H-LIONS, built with 12-port passive AWGR and 5-port active AWGR switches, can use 12 and 5 wavelengths for the intra- and inter- cluster communication respectively. While at the same scale, PETASCALE, built with 12-port switches, can use 18 wavelengths for packet delivery. Thus PETASCALE actually provides better connectivity than H-LIONS.

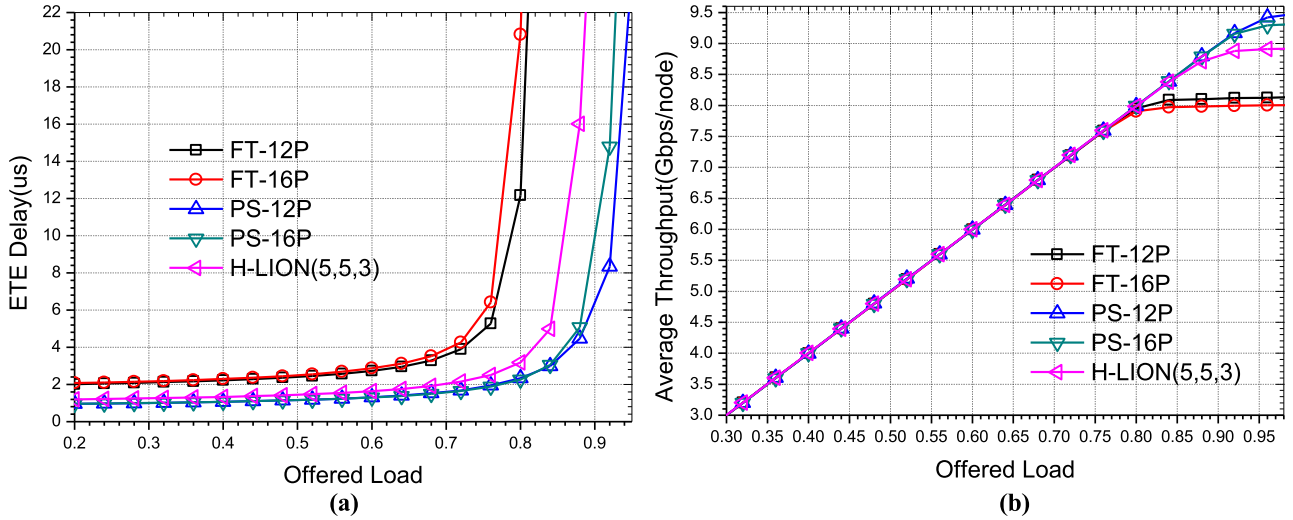


FIGURE 6. The average end-to-end delay (a) and throughput (b) under local traffic pattern.

We then evaluate the network performance under the local traffic pattern. As shown in Fig. 6, fat tree keeps the average end-to-end delay of about 1.9541*us* before it becomes saturated. While PETASCALE and H-LIONS keep a lower unsaturated delays of about 0.9774*us* and 1.2203*us* respectively. This result proves again that OPS networks are able to reduce the end-to-end delay by using the transparent cross connections and limited buffers. For the saturation point, PETASCALE, H-LIONS and fat tree are saturated at 0.90, 0.86, and 0.80 respectively. Apparently, PETASCALE achieves the best performance under the local traffic pattern. This is expected because PETASCALE has provided a rich local connectivity by using *full-mesh* topology and multi-wavelength switching. Moreover, in PETASCALE, the *intra-pod* traffic will only collide at the source switch. This contention can be notified immediately and do not consume the additional bandwidth resource. Thus PETASCALE will become more bandwidth-efficient when more packets are delivered within the local pod.

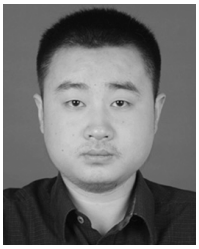
VII. CONCLUSIONS

In this paper, we present PETASCALE, a scalable, low latency, and reliable all-optical interconnects for data center. It exploits the *full-mesh* and *complete bipartite graph* for the intra- and inter- communication respectively, thus supporting scalability beyond 50, 000 servers with much fewer switches and cables. To guarantee a reliable and transparent delivery without the complex and costly buffers, PETASCALE designs the negative acknowledgment and retransmission scheme. Combining with a state retention mechanism, the packet collision can be notified promptly over multiple hops. Then a wavelength-routing algorithm is proposed to restrict the collision domain within two hops from the source server. Moreover, a SOA-based OPS switch is designed to support the multi-hop NACK scheme and wavelength routing. The simulation shows that compared to the electrical packet switching, PETASCALE is able to reduce the average end-to-end delay by more than 50% and deliver about 89% bisection bandwidth of the non-blocking network under uniform traffic pattern. It can even outperform fat tree and other optical designs under the local traffic pattern.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.
[2] R. N. Mysore *et al.*, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM*, 2009, pp. 39–50.
[3] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM*, 2009, pp. 51–62.
[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM*, 2008, pp. 75–86.
[5] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.
[6] A. Singh *et al.*, "Jupiter rising: A decade of CLOS topologies and centralized control in Google's datacenter network," in *Proc. ACM SIGCOMM*, 2015, pp. 183–197.
[7] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2010, pp. 339–350.
[8] G. Wang *et al.*, "c-Through: Part-time optics in data centers," in *Proc. ACM SIGCOMM*, 2010, pp. 327–338.
[9] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, Apr. 2014.
[10] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. ACM SIGCOMM*, 2013, pp. 447–458.
[11] K. Chen *et al.*, "WaveCube: A scalable, fault-tolerant, high-performance optical data center architecture," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 1903–1911.
[12] M. C. Yuang *et al.*, "OPMDC: Architecture design and implementation of a new optical pyramid data center network," *J. Lightw. Technol.*, vol. 33, no. 10, pp. 2019–2031, May 15, 2015.
[13] G. C. Sankaran and K. M. Sivalingam, "Optical traffic grooming-based data center networks: Node architecture and comparison," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1618–1630, May 2016.
[14] V. Kamchevska *et al.*, "Experimental demonstration of multidimensional switching nodes for all-optical data center networks," *J. Lightw. Technol.*, vol. 34, no. 8, pp. 1837–1843, Apr. 15, 2016.
[15] S. Yan *et al.*, "Archon: A function programmable optical interconnect architecture for transparent intra and inter data center SDM/TDM/WDM networking," *J. Lightw. Technol.*, vol. 33, no. 8, pp. 1586–1595, Apr. 15, 2015.

[16] W. M. Mellette *et al.*, "RotorNet: A scalable, low-complexity, optical datacenter network," in *Proc. ACM SIGCOMM*, 2017, pp. 267–280.
[17] H. Rodrigues, R. Strong, A. Akyurek, and T. S. Rosing, "Dynamic optical switching for latency sensitive applications," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, May 2015, pp. 75–86.
[18] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: An overview," *IEEE Commun. Mag.*, vol. 38, no. 2, pp. 84–94, Feb. 2000.
[19] M. J. O'Mahony, D. Simeonidou, D. K. Hunter, and A. Tzanakaki, "The application of optical packet switching in future communication networks," *IEEE Commun. Mag.*, vol. 39, no. 3, pp. 128–135, Mar. 2001.
[20] H. Furukawa, J. M. D. Mendinueta, N. Wada, and H. Harai, "Spatial and spectral super-channel optical packet switching system for multigranular SDM-WDM optical networks," *J. Opt. Commun. Netw.*, vol. 9, no. 1, pp. A77–A84, 2017.
[21] Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, "Recent advances in optical technologies for data centers: A review," *Optica*, vol. 5, no. 11, pp. 1354–1370, 2018.
[22] X. Ye, Y. Yin, S. J. B. Yoo, P. Mejia, R. Proietti, and V. Akella, "DOS: A scalable optical switch for datacenters," in *Proc. 6th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2010, pp. 1–12.
[23] H. Rastegarfar, A. Leon-Garcia, S. LaRochelle, and L. A. Rusch, "Cross-layer performance analysis of recirculation buffers for optical data centers," *J. Lightw. Technol.*, vol. 31, no. 3, pp. 432–445, Feb. 1, 2013.
[24] W. Miao, F. Yan, and N. Calabretta, "Towards petabit/s all-optical flat data center networks based on WDM optical cross-connect switches with flow control," *J. Lightw. Technol.*, vol. 34, no. 17, pp. 4066–4075, Sep. 1, 2016.
[25] F. Yan, W. Miao, O. Raz, and N. Calabretta, "OPSquare: A flat DCN architecture based on flow-controlled optical packet switches," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 4, pp. 291–303, Apr. 2017.
[26] Y. Yin, R. Proietti, X. Ye, C. J. Nitta, V. Akella, and S. J. B. Yoo, "LIONS: An AWGR-based low-latency optical switch for high-performance computing and data centers," *IEEE J. Sel. Topics Quantum Electron.*, vol. 19, no. 2, Mar./Apr. 2013, Art. no. 3600409.
[27] R. Proietti, Z. Cao, C. J. Nitta, Y. Li, and S. J. B. Yoo, "A scalable, low-latency, high-throughput, optical interconnect architecture based on arrayed waveguide grating routers," *J. Lightw. Technol.*, vol. 33, no. 4, pp. 911–920, Feb. 15, 2015.
[28] R. Proietti *et al.*, "Experimental demonstration of a 64-port wavelength routing thin-CLOS system for data center switching architectures," *J. Opt. Commun. Netw.*, vol. 10, no. 7, pp. B49–B57, 2018.
[29] J. Perelló *et al.*, "All-optical packet/circuit switching-based data center network for enhanced scalability, latency, and throughput," *IEEE Netw.*, vol. 27, no. 6, pp. 14–22, Nov. 2013.
[30] G. M. Saridis *et al.*, "Lightness: A function-virtualizable software defined data center network with all-optical circuit/packet switching," *J. Lightw. Technol.*, vol. 34, no. 7, pp. 1618–1627, Apr. 1, 2016.
[31] N. Terzenidis, M. Moralis-Pegios, G. Mourgias-Alexandris, K. Vyrsokinos, and N. Pleros, "High-port low-latency optical switch architecture with optical feed-forward buffering for 256-node disaggregated data centers," *Opt. Express*, vol. 26, no. 7, pp. 8756–8766, 2018.
[32] C. Guo *et al.*, "RDMA over commodity Ethernet at scale," in *Proc. ACM SIGCOMM*, 2016, pp. 202–215.
[33] X. Ye, R. Proietti, Y. Yin, S. J. B. Yoo, and V. Akella, "Buffering and flow control in optical switches for high performance computing," *J. Opt. Commun. Netw.*, vol. 3, no. 8, pp. A59–A72, 2011.
[34] H. Rastegarfar, L. A. Rusch, and A. Leon-Garcia, "WDM recirculation buffer-based optical fabric for scalable cloud computing," *J. Lightw. Technol.*, vol. 32, no. 21, pp. 3451–3465, Nov. 1, 2014.
[35] M. Moralis-Pegios *et al.*, "Multicast-enabling optical switch design employing Si buffering and routing elements," *IEEE Photon. Technol. Lett.*, vol. 30, no. 8, pp. 712–715, Apr. 15, 2018.
[36] C.-H. Wang and T. Javidi, "Adaptive policies for scheduling with reconfiguration delay: An end-to-end solution for all-optical data centers," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1555–1568, Jun. 2017.
[37] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized 'zero-queue' datacenter network," in *Proc. ACM SIGCOMM*, 2014, pp. 307–318.
[38] R. Proietti *et al.*, "All-optical physical layer NACK in AWGR-based optical interconnects," *IEEE Photon. Technol. Lett.*, vol. 24, no. 5, pp. 410–412, Mar. 12, 2012.
[39] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.



XIAOSHAN YU received the M.E. degree in electronics and communications engineering and the Ph.D. degree in telecommunication and information system from Xidian University, in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the State Key Lab of ISN. His main research interests include optical interconnected networks and data center networks.

HUAXI GU received the B.E., M.E., and Ph.D. degrees in telecommunication engineering and telecommunication and information systems from Xidian University, Xi'an, China, in 2000, 2003, and 2005, respectively, where he is currently a Full Professor with the State key lab of ISN, Telecommunication Department. He has more than 100 publications in refereed journals and conferences. His current research interests include interconnection networks, networks on chip, and optical intrachip communication. He has been a Reviewer of the IEEE TRANSACTION ON COMPUTER, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE SYSTEM JOURNAL, the IEEE COMMUNICATION LETTERS, *Information Sciences*, the *Journal of Supercomputing*, the *Journal of System Architecture*, the *Journal of Parallel and Distributed Computing*, and *Microprocessors and Microsystems*.

KUN WANG received the B.E. and M.E. degrees in computer science and technology from Xidian University, Xi'an, China, in 2003 and 2006, respectively, where she is currently a Lecturer with the Department of Computer Science. Her current research interests include high-performance computing, cloud computing, and network virtualization technology.

SHANGQI MA received the B.E. degree in computer and communications from the Lanzhou University of Technology, in 2017. She is currently pursuing the M.E. degree in telecommunication and information systems with the State Key Lab of ISN, Xidian University. Her main research interests include optical interconnection networks and data centers.