Software-Defined Optical Burst Switching for HPC and Cloud Computing Data Centers

Muhammad Imran, Martin Collier, Pascal Landais, and Kostas Katrinis

I. Introduction

Optical networks for data centers have gained significant attention over the past few years due to the potential and benefits of using optical components [1]. The performance of an optical network is directly related to the type of optical switching technique used. These switching techniques are optical circuit switching (OCS), optical packet switching (OPS), and optical burst switching (OBS). OCS is a connection-oriented technique in which a connection is established before actual data transmission on a pre-established dedicated path from the source to the destination [2]. Long connection establishment time and bandwidth underutilizations in the case of low traffic load are the major limitations of OCS. The microelectromechanical system (MEMS) optical cross connect (OXC) or OCS switch has been used in the backbone optical network for many years. Hybrid designs for data center networks (DCNs) that use OCS in conjunction with other technologies have been proposed [3–8]. Helios and c-Through [3,4] propose using OCS in conjunction with traditional electrical packet switching, while the LIGHTNESS project [5,6] employs OCS together with OPS. The Hydra, OSA, and reconfigurable designs [7–9] augment OCS with a multi-hopping technique. A major issue with these interconnects has been their slow reconfiguration time due to the limitation of 3D-MEMS technology. This reconfiguration time is influenced by two factors: 1) the switching time of the 3D-MEMS switch, i.e., 10–100 ms, and 2) the software/control plane overhead required for the estimation of traffic demand and the calculation of a new OCS topology, i.e., 100 ms to 1 s [10]. Consequently, the control plane can only support applications that have high traffic stability, i.e., workloads that last several seconds [3].

In OPS, a packet consists of data and header portions that are in the optical domain. When the packet arrives at the node, the header is removed from the packet and is converted into the electrical domain for processing. During this processing time, the data in the packet has to be buffered in the node. Fiber delay lines (FDLs) are used for this purpose, which can provide limited buffering by routing a light to the specified fibers. The packet is dropped if the switch is not configured within this time. The limitations of OPS are 1) the lack of feasible optical buffers and 2) packet losses in the case of output port contention. OPS for DCNs has been described recently in some studies [10–21].

OPS can only be used with fast optical switching technologies that are now available [12,22–24]. These switches can be built using technologies such as arrayed waveguide grating routers (AWGRs), semiconductor optical amplifiers (SOAs), and $1 \times N$ photonic switches. An AWGR switch is a wavelength router that works in conjunction with tunable wavelength converters (TWCs) or tunable lasers (TLs). An SOA can be used as an optical gate switch. Photonic switches with $1 \times N$ configurations are available and can be used in the Spanke architecture [25] to enable a large switching fabric. AWGR, SOA, and photonic switches have switching times in the range of a few nanoseconds, much faster than the 3D-MEMS switch's tens of milliseconds. Further details about fast optical switches is available in our recent work [26].

OBS [27] has a separate control and data plane similar to OCS. Packets are aggregated into bursts. A control packet is then transmitted on a dedicated control channel to reserve resources on all intermediate nodes from the

source to the destination. The burst is sent at a particular time after sending the control packet, which is called the offset time. During the offset time, these bursts are temporarily stored at edge node before transmission. During this time, the switch controller at the core node processes the control information and sets up the switching matrix for the incoming burst. Burst loss due to output port contention is the major limitation of the OBS network. Several techniques exist in the literature to avoid contention, such as FDLs, deflection routing, wavelength conversion, and segmentation-based dropping, but none of them can guarantee zero burst loss. OBS with two-way reservation ensures zero burst loss in which a control packet reserves resources in all nodes from the source to the destination and is sent back to the source as an acknowledgment. The control packet has a high roundtrip time (RTT) for a large wide area optical network due to high propagation and switching delay.

In our recent work [26], we proposed methods of OBS suitable for DCNs and evaluated their latency. We extend our investigation of these methods in this paper and evaluate throughput and packet loss ratio in addition to latency. We consider diverse traffic workload patterns, various edge-to-core network oversubscription levels, and different data rates. We assess the performance of such designs across a range of usage patterns. We provide a detailed description of our design and evaluate the performance of OBS suitable for use in high performance computing (HPC) and DCNs that makes use of the faster switching technologies that are now available. We implement OBS with a two-way reservation protocol to ensure zero burst loss. The two-way reservation is not suitable for long-haul backbone optical networks due to the high RTT of the control packet. However, for our DCN optical interconnect, this RTT is not high for several reasons: 1) the propagation delay is negligible, 2) faster optical switches are used at the core, 3) a fast optical control plane is used, 4) processing of the control packet is rapid, and 5) a single-hop topology is used. Our techniques exhibit considerable improvement in throughput and packet loss ratio compared to traditional methods of OBS, while delay performance comparable to traditional methods of OBS is also achieved. The proposed technique also demonstrates performance comparable to that of electrical DCNs. A scalability analysis of the proposed design shows that it is scalable to hundreds of thousands of nodes, making it suitable for a very large scale data center.

The rest of the paper is organized as follows. Section II describes the OBS design being modeled. We discuss the scenarios used in performance evaluation in Section III and present the results obtained in Section IV. Section V contains our conclusions.

II. OBS for a Data Center Network

We employ OBS in the proposed data center network architecture. We aggregate packet traffic to create a burst of short duration. A control packet is created to request the allocation of resources needed to transmit the burst from the controller by using a two-way reservation process similar to that proposed for OBS networks [27]. Although such two-way reservation is not feasible in a long-haul backbone network, it is suitable in data centers for the reasons presented earlier. The controller assigns resources and sends the control packet back to the originating node as an acknowledgment. The burst is then transmitted on the pre-established path configured by the controller.

This architecture is shown in Fig. 1(a). The proposed topology has two layers, i.e., the edge and core. The edge contains the electronic top of the rack (ToR) switches, while the core comprises a group of fast optical switches. Servers in each rack are connected to the ToR switches using bidirectional optical fibers. The ToR switches are linked to the optical switches using unidirectional optical fibers.

Our design features separate control and data planes. The control plane comprises a centralized controller that performs routing, scheduling, and switch configuration functions. It receives connection setup requests from all ToR switches, finds routes, assigns timeslots to the connection requests, and configures optical switches with respect to the timeslots allocated. In order to perform these tasks, the controller keeps a record of the connection states of all optical switches. The data plane comprises optical switches that perform data forwarding on pre-configured lightpaths set up by the controller. A management network is used by

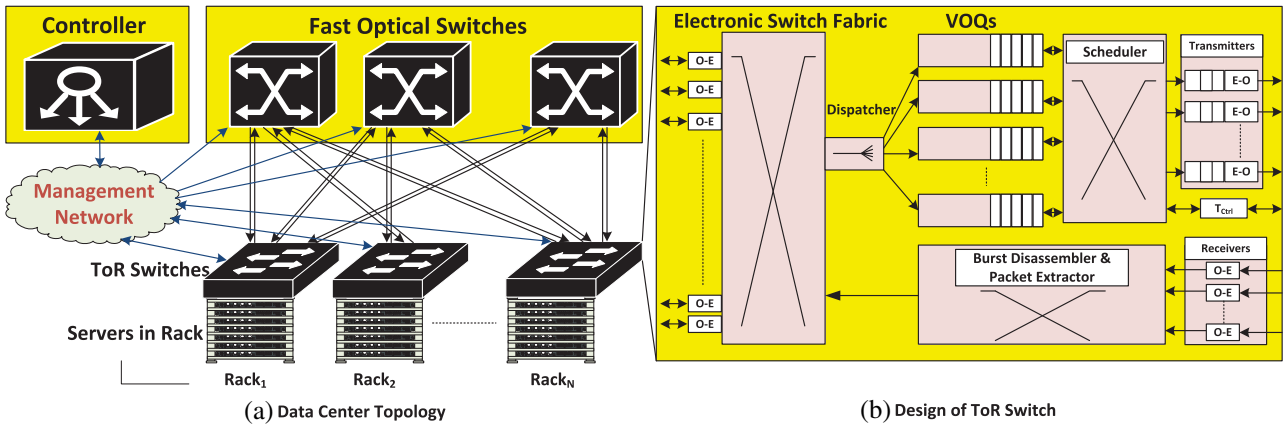

Fig. 1. (a) Topology diagram for a DCN and (b) design of a ToR switch.

the control plane that connects every ToR switch to the controller via a transceiver in each ToR switch reserved for use by the control plane. Further details of this architecture are given elsewhere [28]. OBS with two-way reservation can be more easily implemented in a single-hop topology because the controller has to configure only one optical switch per request.

A. Scalability

A fast optical switch using SOAs as a switching fabric with 1024 ports has been proposed [12], while a fabric with 512 ports using AWGRs as a switching fabric is also feasible [22]. Table I contains a scalability analysis of the proposed topology using both AWGRs and SOAs as the fast optical switches. In Table I, S_{RK} represents the number of servers per rack, while T_{RK} denotes the total number of racks in the DCN. Using SOAs in the switching fabric and ToR switches at its edges, a system size of 40,960 servers with 40 servers per rack or 81,920 servers with 80 servers per rack can be achieved without requiring a multi-stage topology. If we use a pod switch instead of the ToR switch, several ToR switches can be integrated into a single unit, and thus we can aggregate a few hundred to 1000 servers [3]; this makes it feasible to support up to 245,760 servers by considering 240 servers per pod. A system size of 122,880 can also be achieved using AWGRs as a switching fabric by considering 240 servers per pod.

The above analysis considers only data plane issues. Evaluating the performance of the control plane is an open issue as it is dependent on implementation and can only be addressed after deployment of the proposed technique in a real-world scenario. This may be the bottleneck in scaling up the architecture and will be considered in future studies of the architecture. In addition, the feasibility of deployment of multiple controllers could also be investigated.

B. ToR Switch Design

ToR switch design is shown in Fig. 1(b). The ToR switch has an electronic switch fabric that is connected to the servers in the rack to carry out intra-rack (within rack) switching in the electrical domain. To perform inter-rack (between racks) switching, we employ $N-1$ virtual output queues (VOQs), where N is the number of ToR switches in the network. State-of-the-art ToR switches support hundreds of VOQs. For example, the Cisco 5548P supports up to 18,432, and Cisco 5596 supports up to 37,728 VOQs [29,30]. There is a VOQ for each destination ToR switch in the DCN. Packets destined to the same ToR are aggregated into the same VOQ. Each VOQ is configured for a destination network address. Each ToR switch maintains a VOQ table that contains entries of destination rack network addresses and the VOQ number. The dispatcher module matches the destination network address of the packet with the entry in this table and forwards the packet on the required VOQ.

TABLE I
SCALABILITY ANALYSIS

Type	Conf.	S_{RK}	T_{RK}	Servers
SOA	[1024 × 1024]	40	1024	40,960
		80	2048	81,920
		240	6144	245,760
AWGR	[512 × 512]	40	512	20,480
		80	1024	40,960
		240	3072	122,880

C. Control Packet Format

The format of the control packet is shown in Fig. 2. The control packet is 440 bits long and contains two main fields: routing and reservation. The routing field contains the IP address of the source ToR switch, the IP address of the controller, and the IDs of the source and destination ToR switches. We consider 128 bits for IPv6 addresses; however, this length could be reduced to 32 bits using IPv4 addresses, making overall control packet length 31 bytes.

The reservation field is 96 bits long and is divided into three sub-fields: 1) burst length, 2) start time, and 3) port number. The burst length field is filled by the ToR switch to request a timeslot from the controller. The controller fills the other two fields after processing the control packet. All of these three fields are 4 bytes long. The burst length field contains the burst length expressed in bytes; the start time contains the time when the burst will be sent; and the port number is the port of the ToR switch in which the burst is to be sent. A field is reserved for a cyclic redundancy check (CRC), and a couple of optional fields are reserved for flags.

D. Burst Assembly/Disassembly

Burst assembly can be timer-based, length-based, or a combination of both [27]. We consider the mixed approach in which either a timer expires or the burst length exceeds a threshold. The procedure for traffic aggregation is described in Algorithm 1. The timer starts when a packet arrives at the empty VOQ. If the VOQ is not empty when the packet arrives, it joins other packets in the VOQ (lines 11–16 in Algorithm 1). The control packet is generated after the timer expires or the burst length exceeds the threshold and is sent to the controller using a transceiver

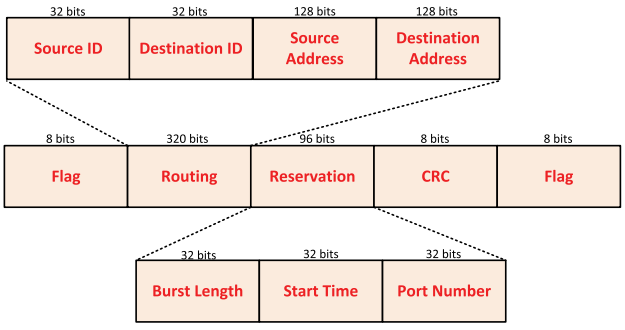

Fig. 2. Control packet format.

dedicated for the control plane (lines 1–8 in Algorithm 1). The control packet at this stage contains information on the burst length, the IP addresses of source and destination ToR switches, and the IDs of source and destination ToR switches. Each ToR switch is assigned a unique ID. The range of IDs of ToR switches is from 0 to $N-1$ in an N rack network. These IDs are used by the controller to perform routing and scheduling algorithms. The controller processes the control packet, assigns start time and port number of the ToR switch on which a burst is to be transmitted, and sends it back to the source ToR switch. The control packet processing mechanism is described in Subsection II.E.

Algorithm 1 Traffic Aggregation at ToR Switch

Require: *timeout* ← *timeoutParameter*
Require: *maxlength* ← *maxBurstLengthParameter*
{timeout and maximum burst length parameters are required during the ToR switches' configuration.}
Require: *timeoutevent* ← *NULL*
Require: *burst_length* ← 0 {Initialize parameters.}
1: **if** *timeoutevent* OR *burst_length* ≥ *maxlength* **then**
2: *control_packet* ← *generateControlPacket*()
3: *control_packet.setBurstLength*(*burst_length*)
4: *control_packet.setSrcId*(*getSrcID*())
5: *control_packet.setDestId*(*getDestID*())
6: *control_packet.setSrcAdd*(*getSrcAdd*())
7: *control_packet.setDestAdd*(*getDestAdd*())
8: *send*(*control_packet*, T_{ctrl})
{timeoutevent is for a timer check. This block is executed when timeoutevent occurs or burst length exceeds the maximum burst length allowed. The control packet is generated and is sent to the management network.}
9: **else**
10: *pk* ← *packetarrives*
11: **if** *VOQ.empty*() **then**
12: *firstpk_time* ← *current_time*
13: *schedule*(*firstpk_time* + *timeout*, *timeoutevent*)
{Schedule timeoutevent by adding timeout parameter in first packet's arrival time.}
14: **end if**
15: *burst_length* += *pk.length*
16: *VOQ.insertPacket*(*pk*)
{**Add** packet in virtual output queue.}
17: **end if**

When the control packet arrives back at the ToR switch, the scheduler module of the ToR switch generates a burst according to the timeslot assigned by the controller. The timeslot refers to the duration of time assigned for a burst in an optical switch path. The generated burst is then sent to the queue of the allocated port. The scheduler module also initiates a new timer if the VOQ is not empty after the burst generation because new packets might have arrived during the RTT of the control packet. The process of burst transmission is explained in Algorithm 2. In the first step, the information of burst size, port number, and start time is extracted from the control packet that arrives at the ToR switch after being processed by the controller (lines 1–4 in Algorithm 2). Next, a burst is generated, and packets are extracted from the VOQ and added into the burst (lines 5–14 in Algorithm 2). The burst is then transmitted from the assigned port number at the assigned timeslot (line 15 in Algorithm 2). After transmitting the burst, the parameters are reinitialized (lines 16–23 in Algorithm 2).

Algorithm 2 Burst Transmission at ToR Switch

Require: *timeout* ← *timeoutParameter*
1: *cp* ← *control packet arrives*
2: *burstsize* ← *cp.getBurstLength*()
3: *portno* ← *cp.getPortNo*()
4: *starttime* ← *cp.getStartTime*()
{Initialization of different variables when the control packet arrives at the ToR switch after being processed by the controller}
5: *burst* ← *generateBurst*()
6: *length* ← 0
7: **while** *VOQ.hasPackets*() **do**
8: **if** *length* ≤ *burstsize* **then**
9: *burst.add*(*VOQ.getPacket*())
10: *length* += *burst.length*
11: **else**
12: *break*
13: **end if**
14: **end while**
15: *sendAt*(*burst*, *portno*, *starttime*)
{The following steps reinitialize variables after burst transmission.}
16: **if** *VOQ* → *empty*() **then**
17: *burst_length* ← 0
18: *firstpk_time* ← 0
19: **else**
20: *pk* ← *VOQ.get*(0)
21: *firstpk_time* ← *pk.arrivaltime*
22: *schedule*(*firstpk_time* + *timeout*, *timeoutevent*)
23: *burst_length* ← *VOQ.getTotalPacketsLength*()
24: **end if**

The ToR switch also has a burst disassembler and packet extractor module to disassemble the bursts received through the receivers. The packets are extracted from the burst and are sent to the electronic switch fabric and finally to the destination servers using electronic switching.

The burst assembly cycle is shown in Fig. 3. Burst assembly time is represented by T_a. The control packet is sent by the ToR switch to the controller for resource

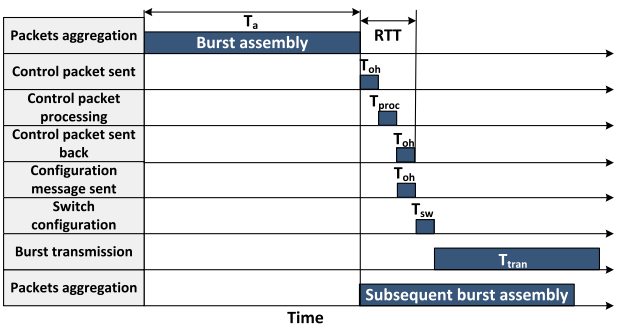

Fig. 3. Burst assembly cycle.

reservation by using a management network. The time that a control packet takes to reach the controller is called overhead time and is represented by T_{oh}. The T_{oh} includes the control packet's propagation delay, optical-electrical-optical (O-E-O) conversion delay, the processing and queuing delay at the electrical switch, and its transmission delay. The controller processes it and assigns a timeslot on an optical switch path. The processing time of the control packet is denoted by T_{proc}. The configuration message also takes T_{oh} to reach the optical switch. T_{sw} is the time that a switch takes to configure an optical switch path and is called switch configuration time. The time a burst takes for the transmission is denoted by T_{tran}. The length of the T_{tran} depends upon the size of the burst and data rate of the channel. As the data rate increases, the length of the T_{tran} decreases for the same size of burst. As soon as the control packet is sent by the ToR switch, a subsequent burst assembly process is also started, and this cycle repeats as long as there is traffic.

E. Control Plane Processing

We consider horizon scheduling that was proposed for the OBS network [27]. The term horizon refers to the latest available time that the channel will be free. Horizon scheduling is explained with the help of Fig. 4. Suppose we have five channels on which an incoming burst can be scheduled. Figure 4(a) shows the states of the channels when a control packet arrives at the controller. The horizon scheduling uses a minimum value method to find the latest available channel. Figure 4(b) shows the states of the channels after allocating a timeslot for the incoming burst.

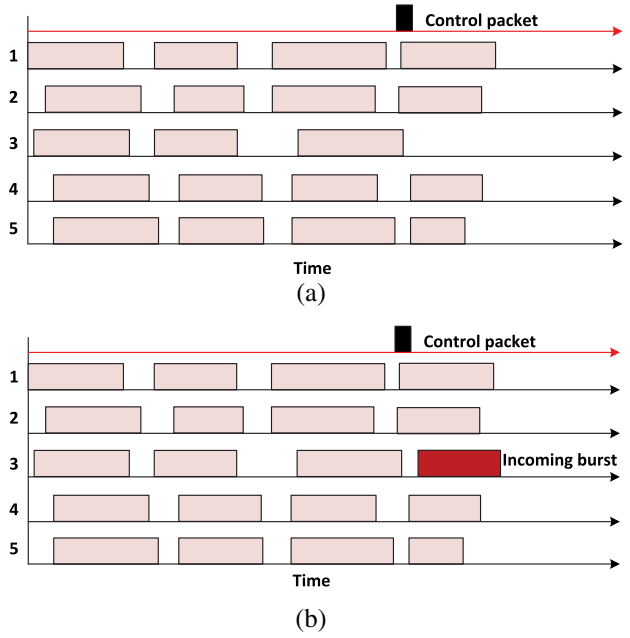

Fig. 4. Resource allocation mechanism using horizon scheduling, i.e., channel states (a) before and (b) after timeslot allocation.

The controller keeps a record of the connections of all optical switches. It performs routing, scheduling, and switch configuration operations. These operations are depicted in Algorithm 3. There are two data structures that are used to maintain the record of horizons of input and output ports (lines 1–2). The controller gets source and destination IDs of ToR switches from the control packet that arrives at the controller (lines 3–5). The controller performs the routing operation by using a technique to find a minimum value method in the input and output horizons (lines 6–30). The routing operation results in finding optimal input/output ports and their relevant horizons.

Algorithm 3 Control Plane Processing

1: $horizoninput[N \times K]$
2: $horizonoutput[N \times K]$
{**Above lines represent data structures of horizons for all inputs and outputs in optical switch paths.**}
3: $control\ packet \leftarrow control\ packet\ arrives\ at\ the\ controller$
4: $srcID \leftarrow controlpacket.getSrcId()$
5: $destID \leftarrow controlpacket.getDestId()$
{**Above lines get source and destination IDs of ToR switches from the control packet that arrives at the controller.**}
6: $minInputHorizon \leftarrow maxValue$
7: $minOutputHorizon \leftarrow maxValue$
8: **for** $i = 0$ **to** $P - 1$ **do**
9: $min1 \leftarrow maxValue$
10: $min2 \leftarrow maxValue$
11: **for** $j = i + srcID \times K$ **to** $(i + srcID \times K + Q - 1)$ **do**
12: **if** $horizoninput[j] < min1$ **then**
13: $min1 \leftarrow horizoninput[j]$
14: $port1 \leftarrow j$
15: **end if**
16: **end for**
17: **for** $k = i + destID \times K$ **to** $(i + destID \times K + Q - 1)$ **do**
18: **if** $horizonoutput[k] < min2$ **then**
19: $min2 \leftarrow horizonoutput[k]$
20: $port2 \leftarrow k$
21: **end if**
22: **end for**
23: $min3 \leftarrow getMax(min1, min2)$
24: **if** $min3 < minInputHorizon$ **and** $min3 < minOutputHorizon$ **then**
25: $minInputHorizon \leftarrow min1$
26: $minOutputHorizon \leftarrow min2$
27: $inputport \leftarrow port1$
28: $outputport \leftarrow port2$
29: **end if**
30: **end for**
{**Above blocks of code select optimal input and output ports and their horizon in the optical switch path.**}
31: $T_{start} \leftarrow getMax(minInputHorizon, minOutputHorizon)$
{**In above line, the maximum of two horizons are assigned to the start time.**}
32: **if** $T_{start} < getCurrentTime()$ **then**
33: $T_{start} \leftarrow getCurrentTime()$
34: **end if**

{**Current time is assigned to the start time if horizons are less than current time.**}

35: $burstlength \leftarrow controlpacket.getBurstLength()$

36: $T_{RL} \leftarrow burstlength * 8/datarate$

{**Requested timeslot T_{RL} is calculated from the burst length (BL) in the control packet.**}

37: $T_{start} \leftarrow T_{start} + T_{sw} + T_{proc} + T_{oh}$

38: $T_{end} \leftarrow T_{start} + T_{RL} + T_{guard}$

{**Above lines represent start and end time of a timeslot in an optical switch path.**}

39: $horizoninput[inputport] \leftarrow T_{end}$

40: $horizonoutput[outputport] \leftarrow T_{end}$

{**Horizons are updated with new time.**}

41: $controlpacket.setstarttime(T_{start})$

42: $controlpacket.setport(inputport \bmod K)$

{**Control packet is updated with start time and port number of the ToR switch.**}

43: $destadd \leftarrow controlpacket.getsourceadd())$

44: $controlpacket.setdestadd(controlpacket.getsourceadd())$

45: $controlpacket.setsourceadd(destadd)$

46: $sendAt(cp, T_{curr} + T_{proc})$

{**Source and destination addresses in the control packet are swapped, and the control packet is sent back to the source ToR. The scheduling operation is completed here. Switch configuration is the next task of the controller.**}

47: $confmsg \leftarrow createConfMsg()$

48: $confmsg.settime(T_{start} - T_{sw})$

49: $confmsg.setinputport((inputport \bmod Q) + (srcID \times Q))$

50: $confmsg.setoutputport((outputport \bmod Q) + (destID \times Q))$

51: $confmsg.setdestadd(getOpticalSwitchAdd(\lfloor \frac{inputport \bmod K}{Q} \rfloor))$

52: $confmsg.setsourceadd(getControllerAdd())$

{**Above lines of code perform switch configuration operation.**}

Scheduling is the next operation that assigns a timeslot on the selected input/output ports (lines 32–46). The length of the timeslot is calculated from the burst length field in the control packet (lines 35–36). The T_{start} and T_{end} represent the start and end time of the timeslot (lines 37–38), respectively. The T_{sw} is the switching time of the optical switch; T_{proc} is the processing time of the control packet at the controller; and T_{oh} is the aggregate time that a control packet spends in the optical plane as discussed earlier. We also consider a guard time T_{guard} in the timeslot to avoid synchronization problems. The horizons on the selected input and output ports are updated with a new time (lines 39–40). The controller updates the control packet by assigning a start time and port number on which the burst will be sent (lines 41–42). It then swaps the source and destination IP addresses in the control packet and sends it back to the source ToR switch (lines 43–46).

Switch configuration is the final task of the controller. After processing the control packet, a configuration message is generated (line 47). The controller sets fields such as input port, output port, and the time at which a switch will be configured. It also fills the source IP address of the controller and the destination IP address of the optical switch. In the end, the configuration message is sent to the switch controller for optical switch configuration. The switch controller configures the optical switch according to the instructions in the configuration message.

III. Performance Analysis

To assess the performance of OBS for a DCN, we developed simulation models in the OMNeT++ simulation framework [31]. This required the control plane algorithms to be implemented in C++ within the OMNeT++ models. The term "software-defined" refers to our own implementation of the control plane in OMNeT++. Porting this code to interact with real switch hardware using, for example, appropriate extensions to the OpenFlow protocol, rather than with the simulation model will allow the control plane to be deployed on generic hardware. The problem of packaging our solution for real-world deployment in a manner compatible with existing SDN frameworks will be addressed in our future work.

A. Network Topology

Our simulation model consists of 40 ToR switches. Each ToR switch has 40 servers connected to it. The controller and ToR switches are connected to the management network via an electrical switch. We use one fast switch that is interfaced to the management network. The two different cases of network oversubscription ratios, i.e., 1:1 and 2:1, are considered to investigate the impact of traffic aggregation on the performance of the system. In a fully subscribed network (1:1), all servers in a rack send their traffic to the servers in other racks, i.e., 100% of the traffic is inter-rack, while in a 2:1 oversubscribed network, 50% of the traffic is inter-rack, and 50% of it is intra-rack.

B. Traffic Generation

To the best of our knowledge, no theoretical model or benchmark of data center traffic has been established yet, but there are a few studies [2,32,33] that have investigated the nature of data center traffic. The *Lognormal* and *Weibull* distributions represent two good models for DCN traffic [33]. We consider *Weibull* distribution for the inter-arrival rate of the packets. Different values for the mean inter-arrival rate of the packets are considered to analyze the performance at different network loads. To represent diversity of traffic workload, we define the term topological degree of communication (TDC). The TDC is the number of simultaneous-destination ToR switches to which a given source ToR switch sends traffic.

C. Simulation Parameters

The key simulation parameters are presented in Table II. We use a value of 1 μs for the switching time of the optical switch because this is a conservative choice, although in some types of fast optical switches, this value can be as low as a few nanoseconds [12,13]. The RTT of the

TABLE II
SIMULATION PARAMETERS

Parameter Name	Symbol	Value
Racks/ToR switches	T_{RK}	40
Servers per rack	S_{RK}	40
Fast optical switch	P	1
Electrical switch for control plane		1
Degree of ToR switches	X	{20,40}
Control packet processing time	T_{proc}	1 μs
Switching time of fast switch	T_{sw}	1 μs
Overhead	T_{oh}	1 μs
Edge-to-core data rate		{10, 40} Gbps
Burst assembly	T_a	{100 μs, 100 KB}, {100 μs, 400 KB}
Topological degree of communication	TDC	{1, 10, 20} racks
Data rate from servers to ToR and for control plane		10 Gbps
Buffer size per port/VOQ		1000 packets

control packet includes its processing time at the controller (T_{proc}) and twice the overhead time (T_{oh}). The aggregate value of T_{oh} is conservatively set to 1 μs. We choose a value of 1 μs for T_{proc}. The value of T_{proc} is compatible with its actual value that we measure in Subsection IV.D. In our recent work [26], we investigate the performance of OBS for data centers using various burst assembly parameters. In this paper, we use optimum values, i.e., {100 μs, 100 KB} for 10 Gbps and {100 μs, 400 KB} for 40 Gbps data rates. We consider three values of the TDC={1,10,20} to investigate the impact of traffic diversity on the performance of the system. We consider a buffer size of 1000 packets (i.e., 1.5 MB) per port/VOQ, while state-of-the-art ToR switches can support a higher buffer size [29,30]. The minimum value of the buffer size should be greater than the maximum burst size (i.e., 100 KB at a 10 Gbps data rate and 400 KB at a 40 Gbps data rate).

IV. RESULTS AND DISCUSSION

We examine the performance of OBS by measuring latency, throughput, and packet loss. We compare the performance of our design using *OBS with two-way reservation* to *OBS using traditional methods of one-way reservation*. We also benchmark the performance against a traditional electrical (TE) packet switching network that features a two-layer leaf-spine topology [1], as shown in Fig. 5. Its performance provides a baseline against which the performance of the new networks can be benchmarked. The simulation results obtained are shown in Figs. 6–10. We also measure and discuss the performance of the algorithms in the control plane later in this section.

A. Latency

We use the term end-to-end delay in order to measure the latency, and we only measure it for inter-rack traffic so that the performance of the optical interconnect could be evaluated. The latency of intra-rack traffic is negligible due to the nanosecond switching times of electrical switches.

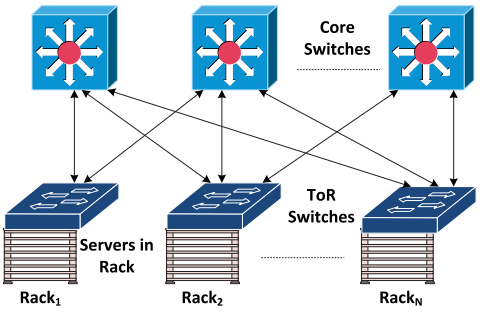

Fig. 5. Topology diagram for the baseline traditional electrical network (leaf-spine topology).

The end-to-end delay is the sum of packet delay incurred at the ToR switch (T_{ToR}), packet transmission delay $\left(\frac{L_{packet}}{B_{core}}\right)$, and the propagation delay (T_{prop}) from the source to the destination servers and is given by the following equation:

$$\text{Delay} = T_{ToR} + \frac{L_{packet}}{B_{core}} + T_{prop}, \quad (1)$$

where L_{packet} is the length of the packet in bits, and B_{core} is the data rate from the ToR switch to the optical switch. The T_{ToR} is the sum of the packet queuing delay at NIC (T_{queue}), packet processing delay (T_{pr}), packet delay for burst assembly ($T_{assembly}$), packet delay until burst departure (T_{depart}), and delay due to O-E-O conversion (T_{oeo}) and is given by the following equation:

$$T_{ToR} = T_{queue} + T_{pr} + T_{assembly} + T_{depart} + T_{oeo}. \quad (2)$$

There is no queuing or processing delay at the optical switch due to all-optical switching.

The simulation results obtained for latency are shown in Figs. 6 and 7. Figure 6 deals with the delay performance at different values of offered load by considering three values for TDC in a fully subscribed network, while Fig. 7 describes the delay performance at different values of offered load by considering three values for TDC in a 2:1 oversubscribed network. The first and second curves of each plot in Figs. 6 and 7 represent end-to-end delay versus offered load using OBS with traditional methods of a one-way reservation scheme using 10 and 40 Gbps data rates, respectively. The third and fourth curves represent the performance of the proposed methods of OBS using a two-way reservation scheme, while the last two curves show the corresponding performance of the baseline electrical network using 10 and 40 Gbps data rates. It can be seen that delay performance in the proposed OBS scheme is compatible with the delay performance of traditional methods of OBS, i.e., the effect of additional delay caused by two-way reservation in the proposed scheme is negligible in all cases of different workloads. However, the delay is a little bit higher in traditional and proposed OBS as compared to the baseline electrical network.

In the baseline electrical network, the delay is around 10 μs until a 90% load and increases thereafter. In a traditional

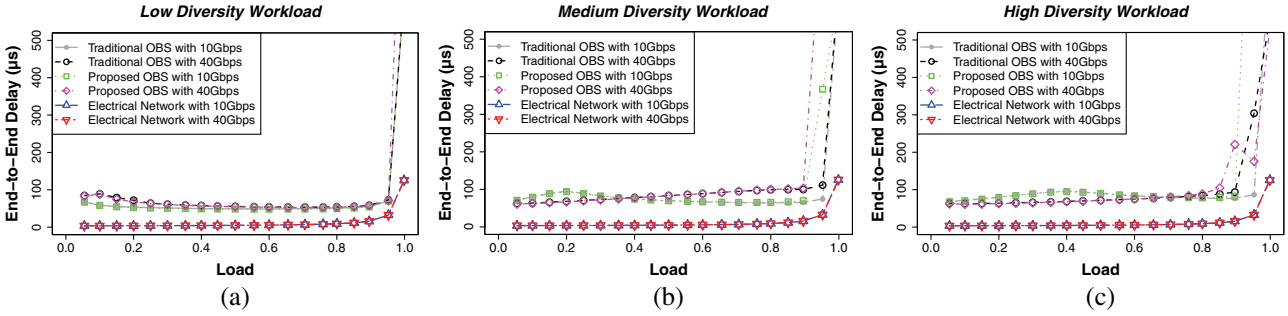

Fig. 6. Load versus end-to-end delay measured in a fully subscribed network for (a) TDC = 1, (b) TDC = 10, and (c) TDC = 20.

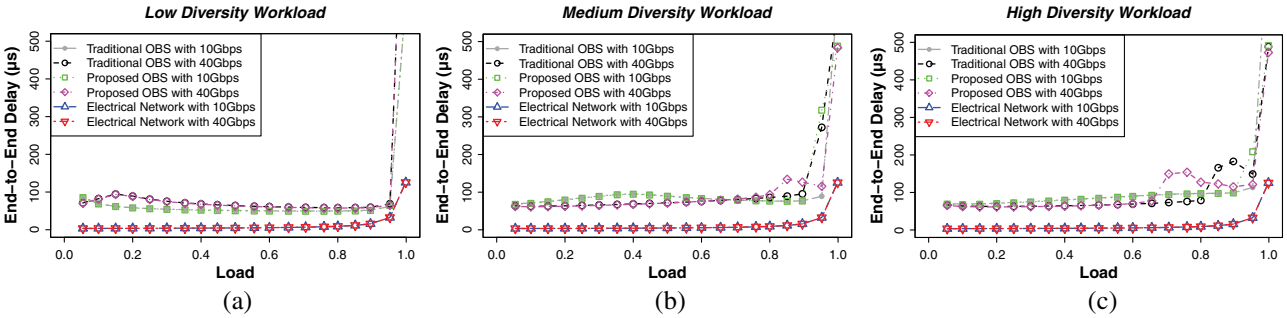

Fig. 7. Load versus end-to-end delay measured with a 2:1 oversubscribed network for (a) TDC = 1, (b) TDC = 10, and (c) TDC = 20.

OBS, the delay is around 50–80 µs up to a high load and increases thereafter. The delay performance of the proposed OBS is similar. The additional delay in OBS is due to the effect of burst assembly delay at the ToR switches, which is an inherent limitation of OBS. Nonetheless, such a delay is acceptable for most HPC applications [34]. HPC applications can be categorized into one of three categories: 1) tightly coupled applications, 2) loosely coupled applications, and 3) parametric execution applications [35]. These applications are characterized by their significant interprocessor communication (IPC) message exchanges among the computing nodes. The tightly coupled applications are very sensitive to latency and require a latency of at most tens of microseconds. In loosely coupled applications, the applications in this category involve little or no IPC traffic among the computing nodes. Thus, low latency is not a requirement. Similarly, the parametric execution applications are also latency insensitive due to the lack of IPC traffic.

B. Throughput

We measure the throughput of each link from the ToR switch to the optical switch in a given amount of time by using the following formula:

$$\mathrm{Th}_{\mathrm{perlink}} = \frac{\mathrm{Total}_{\mathrm{bits}}}{T_{\mathrm{sim}}}, \tag{3}$$

where $\mathrm{Total}_{\mathrm{bits}}$ is the total number of bits successfully delivered through the link in the optical switch, and T_{sim} is the total simulation time. Average network throughput per link is calculated by using the following equation:

$$\mathrm{Th}_{\mathrm{avg}} = \frac{\sum_{n=1}^{N_{\mathrm{link}}} \mathrm{Th}_{\mathrm{perlink}}[n]}{N_{\mathrm{link}}}, \tag{4}$$

where N_{link} is the total number of links of ToR switches connected with optical switches.

The simulation results obtained for throughput are shown in Figs. 8 and 9. Figure 8 presents the throughput performance at different values of offered load by considering three values for TDC in a fully subscribed network, while Fig. 9 exhibits the throughput performance at different values of offered load by considering three values for TDC in a 2:1 oversubscribed network. The first and second curves at each plot in Figs. 8 and 9 represent average throughput per link versus offered load using OBS with traditional methods of a one-way reservation scheme using 10 and 40 Gbps data rates, respectively. The third and fourth curves represent performance of the proposed methods of OBS using a two-way reservation scheme, while the last two curves show the corresponding performance of the baseline electrical network using 10 and 40 Gbps data rates.

Figures 8(a) and 9(a) show that the average throughput is identical in all the networks because the burst loss is zero in OBS with traditional methods until a 95% offered load because bursts are generated in an order and all the bursts are destined to only one destination network. Thus, no collision happens, which results in zero burst loss. However,

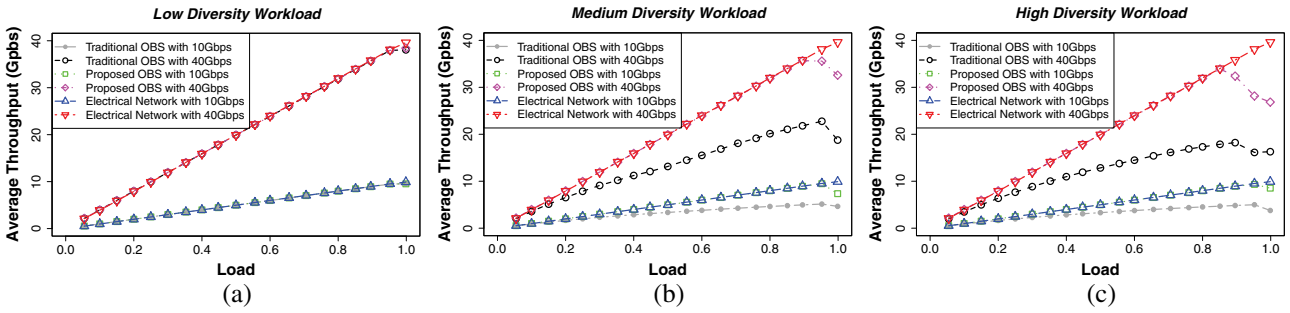

Fig. 8. Load versus average throughput measured in a fully subscribed network for (a) TDC = 1, (b) TDC = 10, and (c) TDC = 20.

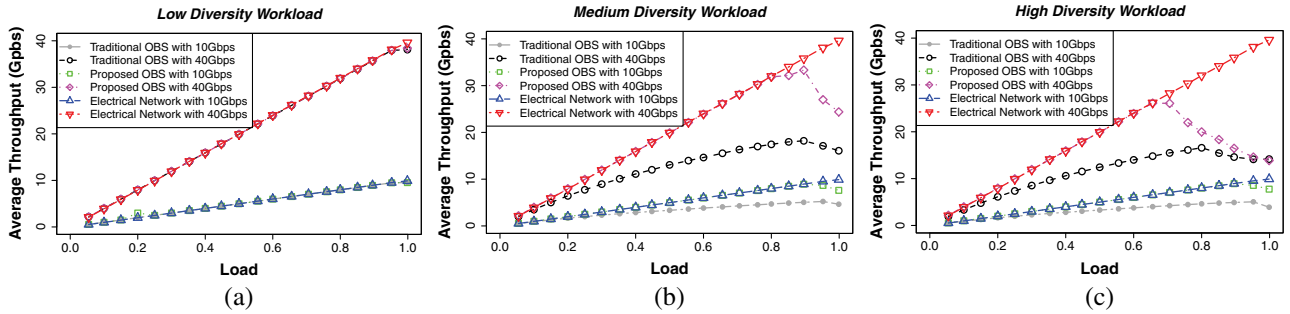

Fig. 9. Load versus average throughput measured with a 2:1 oversubscribed network for (a) TDC = 1, (b) TDC = 10, and (c) TDC = 20.

packets are lost at very high load, i.e., at 95% load, due to buffer overflow at the ToR switches. Packet losses also occur in the proposed scheme as well as in the baseline electrical network due to buffer overflow at a very high load.

In a medium diversity workload as shown in Figs. 8(b) and 9(b), the average throughput in traditional methods of OBS is considerably low as compared to average throughput achieved in the proposed scheme because of burst losses in traditional methods of OBS, whereas in the proposed scheme, burst loss is zero. Due to zero burst loss, our scheme demonstrates comparable performance to the baseline electrical network until a very high load. Another important point is that the average throughput decreases with the increase of data rate at a very high load because bandwidth is wasted during assignment of the timeslot in a link. This wasted bandwidth is 4 times higher in 40 Gbps as compared to 10 Gbps data rates. A similar trend of a decrease in average throughput is also observed with a high diversity workload, as shown in Figs. 8(c) and 9(c).

It is worth noticing the impact of network oversubscription on the average throughput in Figs. 8(b) and 9(b). The drop in average throughput in our proposed scheme with 40 Gbps in a fully subscribed network is less than that of a 2:1 oversubscribed network. A similar trend is observed in Figs. 8(c) and 9(c) because more links are available in a fully subscribed network as compared to a 2:1 oversubscribed network. Thus, the chances of getting a timeslot are high in a fully subscribed network as compared to a 2:1 oversubscribed network. This ultimately results in obtaining a higher average throughput in a fully subscribed network.

C. Packet Loss Ratio

The simulation results obtained for the packet loss ratio are shown in Fig. 10 for different values of offered load. Two values are considered for TDC in a fully subscribed network. The first and second curves in each plot in Figs. 10(a) and 10(b) show the packet loss ratio as a function of offered load for OBS using traditional methods of a one-way reservation scheme at data rates of 10 and 40 Gbps, respectively. The third and fourth curves show the equivalent performance of the proposed methods of OBS using a two-way reservation scheme, while the last two curves show the corresponding performance of the baseline electrical network. Packet losses are observed in the proposed and the baseline network only at a very high load, while the packet losses in traditional OBS occur even at a very low load due to burst losses caused by contention in a traditional OBS. Similar results may be observed in a 2:1 oversubscribed network, although these results are omitted from Fig. 10 for clarity.

D. Performance of the Control Plane

In order to assess the performance of the control plane, we run our algorithm on an Intel host with a Core i7, 2.17 GHz processor and 16 GB RAM. The results were obtained for several combinations of parameters and are shown in Table III.

When a control packet arrives at the controller, the controller performs the routing, scheduling, and switch

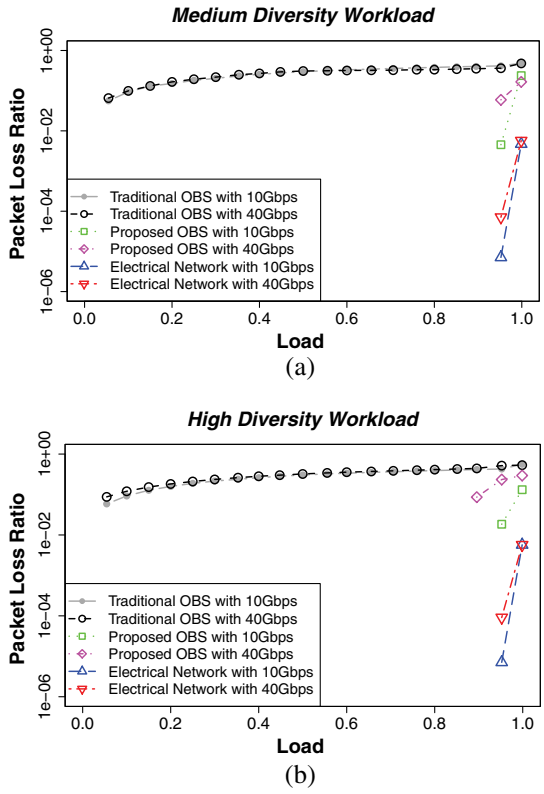

Fig. 10. Load versus packet loss ratio measured in a fully subscribed network for (a) TDC = 10 and (b) TDC = 20.

configuration operations described in Algorithm 3. The routing and scheduling operations are described from line 1–46, and switch configuration operations are described from lines 47–53 in Algorithm 3. The complexity of the routing and scheduling algorithm is $O(2X + \mu)$, where X is the degree of ToR switches and μ represents the sum of processing time of all other instructions. The complexity of the switch configuration operations is $O(P + \mu)$, where P is the total number of optical switches. The μ is assumed to be a constant of negligibly low value. We measure the execution time in fully subscribed, 2:1 oversubscribed, and 4:1 oversubscribed networks, as shown in Table III. We assume 40 servers per rack. It can be noticed in Table III that the execution time of routing and scheduling operations is in a nanoseconds scale for all types of networks. Execution time is the lowest in the 4:1 oversubscribed network, but it increases slightly as we decrease network oversubsription. Similarly, the execution time of the switch configuration operations is at a minimum when P is minimum, and it increases slightly with an increase in the number of optical switches. The overall execution time of switch configuration operations is negligible (at most a few nanoseconds). We obtain total execution time of the control plane processing by adding up the execution times of routing/scheduling and switch configuration operations, which is in the nanoseconds range. Thus, our algorithms in the control plane demonstrate efficient performance for all types of network oversubscriptions.

TABLE III
PERFORMANCE OF THE CONTROL PLANE

Algorithm	Co	P	X	Exec. Time (μs)
Routing and scheduling	4:1	$\forall P$	10	<0.15
	2:1		20	<0.3
	1:1		40	<1
Switch configuration	4:1	10	10	≈0.029
	2:1	20	20	≈0.031
	1:1	40	40	≈0.033

V. CONCLUSION

We proposed a novel optical interconnect based on fast optical switches. The proposed design features fast optical switches in a single-hop topology with a centralized, software-defined optical control plane. The single-stage core topology can be easily scaled up (in capacity) and scaled out (in the number of racks) without requiring major recabling and network reconfiguration.

We use OBS with two-way reservation to obtain zero burst loss. Two-way reservation is not appropriate for conventional backbone optical networks due to the high RTT of the control packet, but in a DCN, the RTT is not high. We use network-level simulation to model different workloads with various data rates by considering different edge-to-core network oversubscription and investigate the performance of such designs across usage patterns. Our results reveal that the proposed technique shows considerable improvement in terms of throughput and packet loss ratio as compared to conventional methods of OBS, while comparable performance in terms of delay with conventional methods of OBS is also achieved. The proposed technique also demonstrates performance comparable to that of electrical data center networks.

ACKNOWLEDGMENT

The work was supported by the Irish research council (IRC) through the Enterprise Partnership Scheme and IBM Ireland. We are also grateful to the National Centre for Physics, Pakistan, for providing access to the high speed servers for running simulations.

REFERENCES

[1] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 4, pp. 1021–1036, 2012.

[2] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *Proc. ACM/IEEE Conf. on Supercomputing*, 2005, p. 16.

[3] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 339–350, 2011.

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

[5] W. Miao, F. Agraz, S. Peng, S. Spadaro, G. Bernini, J. Perelló, G. Zervas, R. Nejabati, N. Ciulli, D. Simeonidou, H. Dorren, and N. Calabretta, "SDN-enabled OPS with QoS guarantee for reconfigurable virtual data center networks," *J. Opt. Commun. Netw.*, vol. 7, no. 7, pp. 634–643, 2015.

[6] S. Peng, B. Guo, C. Jackson, R. Nejabati, F. Agraz, S. Spadaro, G. Bernini, N. Ciulli, and D. Simeonidou, "Multi-tenant software-defined hybrid optical switched data centre," *J. Lightwave Technol.*, vol. 33, no. 15, pp. 3224–3233, 2015.

[7] K. Christodoulopoulos, D. Lugones, K. Katrinis, M. Ruffini, and D. O'Mahony, "Performance evaluation of a hybrid optical/electrical interconnect," *J. Opt. Commun. Netw.*, vol. 7, no. 3, pp. 193–204, March 2015.

[8] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, April 2014.

[9] D. Lugones, K. Katrinis, G. Theodoropoulos, and M. Collier, "A reconfigurable, regular-topology cluster/datacenter network using commodity optical switches," *Future Generation Computer Systems*, vol. 30, pp. 78–89, 2014.

[10] G. Porter, R. Strong, N. Farrington, A. Forencich, C.-S. Pang, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 447–458, 2013.

[11] O. Liboiron-Ladouceur, P. G. Raponi, N. Andriolli, I. Cerutti, M. S. Hai, and P. Castoldi, "A scalable space–time multiplane optical interconnection network using energy-efficient enabling technologies [Invited]," *J. Opt. Commun. Netw.*, vol. 3, no. 8, pp. A1–A11, 2011.

[12] O. Liboiron-Ladouceur, I. Cerutti, P. G. Raponi, N. Andriolli, and P. Castoldi, "Energy-efficient design of a scalable optical multiplane interconnection architecture," *IEEE J. Sel. Top. Quantum Electron.*, vol. 17, no. 2, pp. 377–383, 2011.

[13] Y. Yin, R. Proietti, X. Ye, C. J. Nitta, V. Akella, and S. Yoo, "LIONS: An AWGR-based low-latency optical switch for high-performance computing and data centers," *IEEE J. Sel. Top. Quantum Electron.*, vol. 19, no. 2, 3600409, 2013.

[14] R. Proietti, Z. Cao, C. Nitta, Y. Li, and S. Yoo, "A scalable, low-latency, high-throughput, optical interconnect architecture based on arrayed waveguide grating routers," *J. Lightwave Technol.*, vol. 33, no. 4, pp. 911–920, Feb. 2015.

[15] G. Wu, H. Gu, K. Wang, X. Yu, and Y. Guo, "A scalable AWG-based data center network for cloud computing," *Opt. Switching Netw.*, vol. 16, pp. 46–51, 2015.

[16] P. N. Ji, D. Qian, K. Kanonakis, C. Kachris, and I. Tomkos, "Design and evaluation of a flexible-bandwidth OFDM-based intra-data center interconnect," *IEEE J. Sel. Top. Quantum Electron.*, vol. 19, no. 2, 3700310, 2013.

[17] O. Liboiron-Ladouceur, A. Shacham, B. A. Small, B. G. Lee, H. Wang, C. P. Lai, A. Biberman, and K. Bergman, "The data vortex optical packet switched interconnection network," *J. Lightwave Technol.*, vol. 26, no. 13, pp. 1777–1789, 2008.

[18] Q. Yang, "Latency-optimized high performance data vortex optical switching network," *Opt. Switching Netw.*, vol. 18, pp. 1–10, 2015.

[19] A. S. Hamza, J. S. Deogun, and D. R. Alexander, "Free space optical multicast crossbar," *J. Opt. Commun. Netw.*, vol. 8, no. 1, pp. 1–10, 2016.

[20] T. Ismail, "Optical packet switching architecture using wavelength optical crossbars," *J. Opt. Commun. Netw.*, vol. 7, no. 5, pp. 461–469, May 2015.

[21] S. Liu, Q. Cheng, M. R. Madarbux, A. Wonfor, R. V. Penty, I. H. White, and P. M. Watts, "Low latency optical switch for high performance computing with minimized processor energy load [Invited]," *J. Opt. Commun. Netw.*, vol. 7, no. 3, pp. A498–A510, March 2015.

[22] K. Takada, M. Abe, M. Shibata, M. Ishii, and K. Okamoto, "Low-crosstalk 10-GHz-spaced 512-channel arrayed-waveguide grating multi/demultiplexer fabricated on a 4-in wafer," *IEEE Photon. Technol. Lett.*, vol. 13, no. 11, pp. 1182–1184, 2001.

[23] S. Aleksic, "Analysis of power consumption in future high-capacity network nodes," *J. Opt. Commun. Netw.*, vol. 1, no. 3, pp. 245–258, 2009.

[24] K. Nashimoto, D. Kudzuma, and H. Han, "High-speed switching and filtering using PLZT waveguide devices," in *15th OptoElectronics and Communications Conf. (OECC)*, 2010, pp. 540–542.

[25] J. Aracil and F. Callegati, *Enabling Optical Internet with Advanced Network Technologies*. Springer, 2009.

[26] M. Imran, M. Collier, P. Landais, and K. Katrinis, "Performance analysis of optical burst switching with fast optical switches for data center networks," in *17th IEEE Int. Conf. on Transparent Optical Networks*, Hungary, 2015.

[27] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: A new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16–23, 2004.

[28] M. Imran, M. Collier, P. Landais, and K. Katrinis, "Performance evaluation of hybrid optical switch architecture for data center networks," *Opt. Switching Netw.*, vol. 21, pp. 1–15, July 2016.

[29] "Cisco Nexus 5596" [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5548p-switch/white_paper_c11-622479.html.

[30] "Cisco Nexus 5548P, 5548UP, 5596UP, and 5596T switches data sheet" [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5000-series-switches/data_sheet_c78-618603.html.

[31] "OMNeT++ simulation framework" [Online]. Available: http://omnetpp.org/.

[32] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Measurement*, 2009, pp. 202–208.

[33] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. on Internet Measurement*, 2010, pp. 267–280.

[34] J. Heo and L. Singaravelu, "Deploying extremely latency-sensitive applications in vSphere 5.5: Performance study," Technical Report, VMware, Inc., 2013 [Online]. Available: www.vmware.com/files/pdf/techpaper/latency-sensitive-perf-vsphere55.pdf.

[35] "Cut-through and store-and-forward Ethernet switching for low-latency environments" [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-465436.html.