Dynamic Topology Management in Optical Data Center Networks

Yangming Zhao, Sheng Wang, *Member, IEEE*, Shouxi Luo, Vishal Anand, Hongfang Yu, Xiong Wang, Shizhong Xu, and Xiaoning Zhang

***Abstract*—In recent years, there has been an exponential growth in the traffic of cloud data centers networks (DCNs). More and more users are outsourcing their computing demands and services to DCNs. To handle such increases in the workload and traffic in a scalable manner, many technologies have been proposed to provide dynamic topology to the network so as to adapt to the traffic. One representative scheme is the optical switching architecture (OSA) as it provides tremendous flexibility. However, OSA requires too much time (10 ms) to reconfigure the topology. Reconfiguring the topology in one step not only brings much performance degradation to the delay sensitive flows, but also incurs large traffic losses in high throughput low latency DCNs. Therefore, a progressive topology reconfiguration scheme is required to guarantee the performance of delay sensitive flows and reduce the traffic loss. To this end, we propose a topology management algorithm (TMA) in this paper to design such a progressive topology reconfiguration scheme. With TMA, a sequence of intermediate topologies and routing schemes are derived. We leverage these intermediate topologies to realize the reconfiguration progressively, while maintaining the topology connectivity and reducing the total traffic loss during the topology reconfiguration. Through extensive real-trace driven simulation experiments, we find that TMA can reduce the traffic loss during topology reconfiguration with little overhead.**

***Index Terms*—Data center networks (DCNs), intermediate topology, optical switching architecture (OSA).**

I. Introduction

The growing significance of data or "Big Data" and the increased interest and need to analyze these large amounts of information has led to the building of large scale data centers. Without these data centers it is impossible to analyze this huge amount of information by using just traditional database management tools or data processing applications. These data centers pool computational, storage and network resources using a communication network. Data center networks (DCNs) play a central role by interconnecting all of the data center resources together. The growing quantity of data and the popularity of data centers has led more users to outsource their large computing demands to cloud data centers, which has led to a significant increase in the traffic of DCNs. There are usually two ways to handle such dramatic growth in the traffic volume of DCNs. The first one is applying intelligent virtual machine (VM) placement algorithms to allocate VMs with high traffic demands to proximate hosts with high bandwidth connectivity, while the other one is to design a topology with high bisection bandwidth and rich connectivity. In this paper, we focus on the second option to deal with the dramatic traffic growth.

To design a topology with high connectivity and rich bisection bandwidth, one approach is statically provisioning high bandwidth and rich connectivity to every pair of hosts as in a Fattree [1], VL2 [2], BCube [3], DCell [4] and FSO-DC [5] architectures. In many real traffic traces, however, we see that the traffic matrices in DCNs are time-varying and there is little to no traffic in some parts of the network. Furthermore, workload consolidation policies may cause traffic imbalance. This implies that such a network may be underutilized and significant amount of network resources are wasted. To overcome this shortage, wireless communication technologies (e.g., Flyways [6] and 3-D beamforming [7]) or optical switching technologies (e.g., OSA [8] , Helios [9] and c-Through [10]) are exploited to design dynamic topologies for DCNs. With these technologies, the network topology can be dynamically configured according to the traffic matrix, and hence the network resource utilization is enhanced. At a time when DCN size and traffic are increasing exponentially, dynamic topology technology is the most promising to realize DCN interconnection.

Among the dynamic topology technologies, optical switching architecture (OSA) [8] is the state-of-the-art work. OSA can form any D-regular topology (where D is a predefined number according to the module connection) by changing the configuration of the micro-electro-mechanical switch (MEMS), and adjusting the capacity of links by changing the number of wavelengths assigned to it. Therefore, OSA provides the most flexibility to the network topology to adapt to the traffic, and enhance network resource utilization. On the other hand, the OSA leverages the optical components to reduce the energy consumption, which is important in DCNs and motivates the operators to adopt OSA (see technology details of OSA in Section II-A).

Though OSA can greatly improve the network resource utilization, it brings more network management complexity, such

Manuscript received March 8, 2015; revised May 19, 2015 and July 7, 2015; accepted July 29, 2015. Date of publication August 2, 2015; date of current version August 17, 2015. This paper was presented in part at the IEEE Global Telecommunications Conference, Austin, TX, USA, December 8–12, 2014. This work was supported in part by the 973 Program under Grant 2013CB329103, NSFC Fund under Grants 61271165, 61271171, 61201129, and 61301153, Program for Changjiang Scholars and Innovative Research Team in the University, the 111 Project B14039, and the Fundamental Research Funds for the Central Universities. *(Corresponding author: Sheng Wang).*

Y. Zhao, S. Wang, S. Luo, H. Yu, X. Wang, S. Xu, and X. Zhang are with the Key Lab of Optical Fiber Sensing and Communication, Education Ministry of China, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: zhaoyangming@uestc.edu.cn; wsh_keylab@uestc.edu.cn; rithmns@gmail.com; yuhf@uestc.edu.cn; wangxiong@uestc.edu.cn; xsz@uestc.edu.cn; xnzhang@uestc.edu.cn).

V. Anand is with the Department of Computer Science, College at Brockport, The State University of New York, Brockport, NY 14420 USA (e-mail: vanand@brockport.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2015.2464079

as how to change the network topology, link capacity and traffic routing from the old to the new settings, based or optimized according to the traffic variation. One simple method is directly reconfiguring the OSA to the desired settings in one step. However, this scheme may hurt the performance of DCNs. Previous measurement results [11] show that 80% of the flows in the datacenter are smaller than 10 kB, but most of the bytes are in the 10% of elephant flows. In high speed and low latency DCNs, the mice flows smaller than 10 kB complete in 1 or 2 RTTs which is less than 1 ms [12]. Thus, waiting for the OSA to reconfigure (about 10 ms [8] to reconfigure the entire system) is too large an overhead for the mice flows. Even worse, some of the mice flows have deadlines, and 10 ms is a very common target to control application latency in DCNs [13]. Waiting for the OSA reconfiguration prevents these deadline-aware flows from meeting their deadline requirements. On the other hand, directly reconfiguring all the network interconnection and link capacity may cause a large amount of traffic loss for the elephant flows. Suppose there are 80 top-of-racks (ToRs) in an OSA-based DCN such that each ToR supports 32 wavelengths and the bandwidth of each wavelength is 10 Gb/s, 10 ms for OSA reconfiguration may incur 2.56 Gb traffic loss in the worst case.

To solve the problems discussed above, we propose a progressive reconfiguration approach for the OSA-based DCNs, such that all the mice flows that are delay sensitive can be served continuously and the traffic loss during reconfiguration can be reduced. To constantly serve all the mice flows, we should maintain the network connectivity during the configuration and temporarily route the mice flows through the unchanged part. Furthermore, to reduce the traffic loss, we should determine the traffic routing and wavelength assignment during reconfiguration to fully utilize the network resources.

Accordingly, given the initial and the final (or target) network topology, wavelength assignment and traffic routing, the progressive reconfiguration method should determine 1) link reconfiguration order, i.e., the intermediate topologies to complete the reconfiguration; 2) wavelength assignment on the intermediate topologies; and 3) traffic routing on each intermediate topology. To the best of our knowledge, our work is the first to design such progressive reconfiguration approach in OSA-based DCNs. Though such reconfiguration includes both network interconnection and wavelength assignment, for simplicity we use the term topology reconfiguration in this paper.

The main contributions of our work can be summarized as follows:

- We formulate the topology reconfiguration problem (TRP) in OSA-based DCNs, and analyze the complexity and feasibility of the problem.
- Based on our analysis, we propose a topology management algorithm (TMA) to guarantee the network connectivity and reduce traffic loss during the reconfiguration.
- We also discuss how to implement TMA in real DCNs.
- We evaluate our analysis and the TMA algorithm through detailed simulation experiments and show that our approach can greatly reduce the traffic loss during topology reconfiguration with only little overhead to the system.

Compared with the conference version, this extended version contains:

TABLE I
NOTATION USED IN OUR WORK

Notation	Description
$\boldsymbol{G}_i$	$i = 0, 1, 2 \ldots, n$: The network topology for the i^{th} intermediate topology, with $\boldsymbol{G}_0$ being the initial topology and $\boldsymbol{G}_n$ being the final topology. In addition, we call $\boldsymbol{G}_{i-1}$ and $\boldsymbol{G}_i$ as *adjacent topologies* for all $i = 1, 2, \ldots, n$. $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ is the *common part* of $\boldsymbol{G}_{i-1}$ and $\boldsymbol{G}_i$. The common part will be unchanged when we convert the topology from $\boldsymbol{G}_{i-1}$ to $\boldsymbol{G}_i$.
$\Delta(\boldsymbol{G})$	The maximum nodal degree in $\boldsymbol{G}$.
P^i_{sd}	$i = 1, 2 \ldots, n$: The set of edges used by the path from s to d in $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$.
W^{ki}_{uv}	Binary variable whose value is 1 if wavelength k is assigned to link (u, v) in $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$.
C	Constant that denotes the bandwidth of a wavelength in the system.
D	Constant that denotes the maximum nodal degree of a ToR in the OSA system.
W_N	Constant that denotes the number of available wavelengths in the OSA system.
M	Constant that denotes the number of MEMS ports in the OSA system.
f_{sd}	Constant that denotes the required traffic rate from ToR s to ToR d.
N	Constant that denotes the number of ToRs in the network.
r^i_{sd}	The real traffic rate from ToR s to ToR d in $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$.
l^{sdi}_{uv}	The traffic rate from ToR s to ToR d that is carried by edge (u, v) in $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$.
e^i_{uv}	Binary variable whose value is 1 if $(u, v) \in \boldsymbol{G}_i$, and 0 otherwise.
a^i_{uv}	Binary variable whose value is 1 if $e^{i-1}_{uv} = 1$ and $e^i_{uv} = 1$, and 0 otherwise.
c^{sdi}_{uv}	Auxiliary binary variable whose value is 1 if (u, v) is used to maintain the connectivity from ToR s to ToR d when topology $\boldsymbol{G}_{i-1}$ is converted to $\boldsymbol{G}_i$, and 0 otherwise.

- A detailed motivation example to show why progressive reconfiguration method is helpful.
- A detailed and full programming formulation for the problem.
- A detailed discussion on how to implement TMA in the real DCNs.
- More detailed simulations to evaluate the performance of TMA and an evaluation of the system overhead due to the use of TMA.

In Section II, we introduce the OSA system which is fundamental to our work, and show a motivation example on how progressive topology reconfiguration benefits dynamic DCNs. We then formulate and analyze this TRP in Section III-B. Based on our analysis, an algorithm for progressive topology reconfiguration named TMA is proposed in Section IV. Extensive simulation results are presented to evaluate the performance of TMA in Section V followed by some related works in Section VI. At last, we conclude this paper in Section VII. For clarity, we list all the notations that we use in this paper in Table I.

II. Background and Motivation

In this section, we first briefly review the OSA and show how progressive topology reconfiguration benefits OSA-based DCNs through an example.

A. *Optical Switching Architecture*

Fig. 1(a) shows an abstraction of an OSA [8]. For simplicity, we ignore details of the technology and only show those aspects that are most related to our work. In this system, all traffic from hosts under a ToR is converted to optical signals on multiple

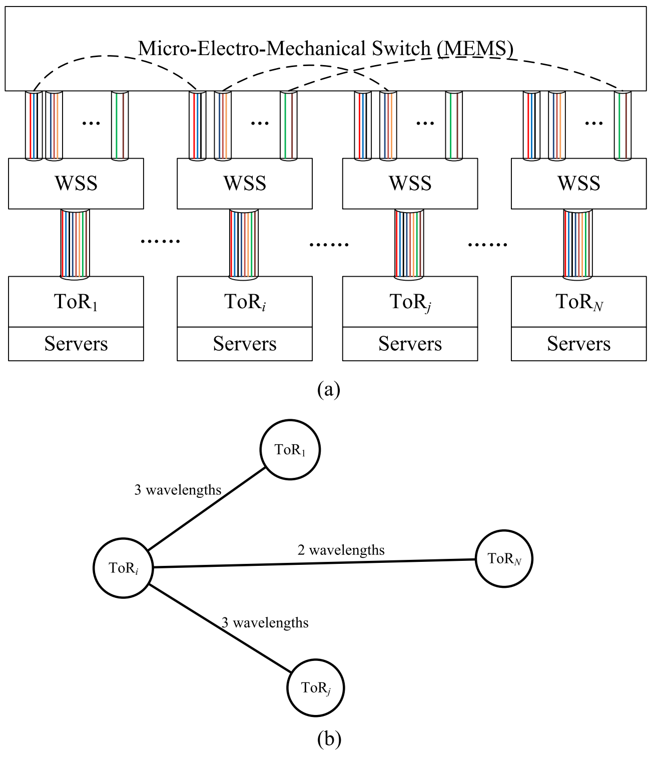


Fig. 1. OSA. (a) OSA abstraction. (b) Logic connection of ToR i.

wavelengths, and then multiplexed into a fiber connected to a wavelength selective switching (WSS) module. The WSS divides all these wavelengths into D groups and sends them into D different MEMS ports, so that these wavelengths may be switched to other ToRs.

The flexibility of the OSA in realizing dynamic DCN topologies is achieved by exploiting the reconfigurability of the MEMS. MEMS is one of the most popular technologies for matrix optical switching that allow simultaneous connection between a number of input and output fibers, in a fully non-blocking all-optical manner. It achieves one-to-one circuits by using electromechanically controlled micro-mirrors to steer optical beams. In other words, we can logically connect any two MEMS ports directly, in turn also connecting the two ToRs associated with these two ports. Take ToR_i in Fig. 1(a) as an example, three of the MEMS ports associated with it are connecting to the ports used by ToR_1, ToR_j and ToR_N, respectively. Logically, ToR_i is directly connecting with these three ToRs as shown in Fig. 1(b). If we connect each of N ToRs to D ports of MEMS (assume $D \times N = M$), it means that each ToR is directly connecting with D ToRs, i.e., the logical topology of OSA system is a D-regular graph. By leveraging the reconfigurable characteristic, OSA can form all the D-regular topologies for DCNs.

Another advantage of the OSA is that it can flexibly modify the capacity of each link on the logical topology by controlling the WSS. If two ToRs want to communicate with capacity k times of a single wavelength, it can be easily achieved by WSS splitting k wavelengths to the appropriate port. Again, take the case shown in Fig. 1 as an example, if both the WSSes connecting with these two ToRs in Fig. 1(a) should split the *same* three wavelengths to the appropriate ports that are connected by MEMS, the logic link capacity between ToR_i and ToR_j is three times of a single wavelength as shown in Fig. 1(b). By leveraging the real time reconfiguration property of WSS, OSA can dynamically adjust the logical link capacity. For simplicity, we use the terms topology and link to imply logical topology and logical link, respectively.

For example, according to [8], for a network with 2560 severs, OSA consumes almost the same energy compared with the traditional 2:1 over-subscribed network (the power consumption of both networks are 73 kW), which can only achieve 50% non-blocking switching, while OSA achieves 60%–100% non-blocking switching. Furthermore, compared to the Fattree structure which can achieve 100% non-blocking switching, OSA consumes less than half of the energy that consumed by Fattree (196 kW).

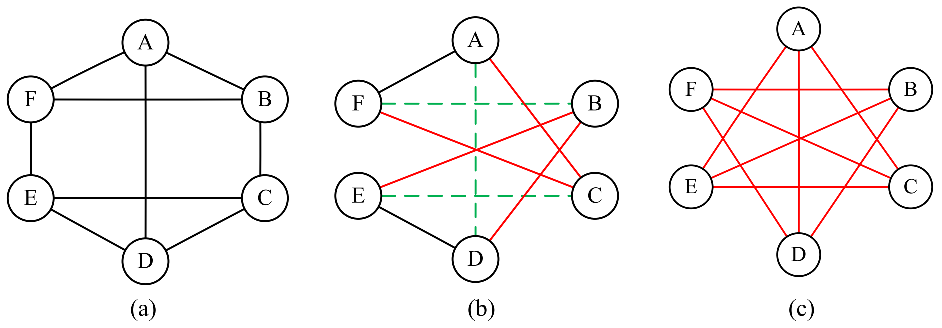
B. Motivation Example

In this section, we use an example to show why progressive topology reconfiguration can help for topology reconfiguration in OSA-based DCNs. The example is shown in Fig. 2, where each node represents a ToR and the capacity of each link is 1 Gb/s (bidirectional). Suppose that in the first time instance, the network topology in Fig. 2(a) can yield the best network performance (e.g., minimum used network capacity). However, in the second time instance, the traffic matrix varies to be the case shown in Fig. 2(d), and hence, the network topology should be reconfigured as shown in Fig. 2(c) to reduce the used network capacity.

A simple approach to transform the topology from Fig. 2(a) to (c) is straightforwardly reconfiguring the MEMS and WSSes according to the interconnection shown in Fig. 2(c). If so, only three links (shown as green dotted lines in Fig. 2(b)) in the network are maintained and can be used to carry traffic during the reconfiguration process. In this case, at least 60 Mb traffic is lost if 10 ms is required for the reconfiguration. Even worse, the network is broken into several disconnected partitions during the reconfiguration and all the cross-partition traffic will be dropped. Waiting for 10 ms is too large an overhead for the delay-sensitive mice flows.

An alternative scheme is to progressively complete such topology reconfiguration in three steps. First, we route each flow through the *Route 1* shown in Fig. 2(d) and configure the topology to the one shown in Fig. 2(b). All the links that are carrying traffic are maintained and there is no overflow on these links. Thus, no traffic loss occurs. In the second step, we change the flow route to the *Route 2* presented in Fig. 2(d) and configure the topology to be the one shown in Fig. 2(c). Again, no links carrying traffic are removed and no traffic loss occurs. Lastly, we route all the traffic to the paths calculated by an optimal algorithm. By introducing an intermediate topology as shown in Fig. 2(b) and temporarily routing traffic as shown in Fig. 2(d), we realize the traffic-loss free topology reconfiguration.

We note that we cannot always realize topology reconfiguration without traffic loss, e.g., if the traffic rate of each flow in



Source	A	A	A	B	B	B	C	C	D
Sink	C	D	E	D	E	F	E	F	F
Rate (Mbps)	100	100	100	100	100	100	100	100	100
Route 1	A→D→E→C	A→D	A→D→E	B→F→A→D	B→F→A→D→E	B→F	C→E	C→E→D→A→F	D→A→F
Route 2	A→C	A→D	A→C→E	B→D	B→E	B→F	C→E	C→F	D→B→F

(d)

Fig. 2. Motivation Example. (a) Initial topology. (b) Intermediate topology. (c) Final topology. (d) New traffic matrix and temporary routing.

Fig. 2(d) is 1 Gb/s, there are no means to fix all the flows during the topology reconfiguration due to the lack of network resource. However, if we can maintain a connected topology during the reconfiguration, we can at least guarantee the performance of delay sensitive flows by temporarily routing them through the connected part that is maintained during the configuration.

Thus, by leveraging the intermediate topologies to maintain a connected part unchanged when we change one topology to another, we expect to guarantee the performance of delay sensitive flows at first. Then, with appropriate intermediate routing and wavelength assignment, we reduce the traffic loss during the topology reconfiguration, even though sometimes, traffic loss is inevitable due to the lack of network resource.

III. Problem Formulation and Analysis

In this section, we formulate the TRP as a mathematic model and then analyze the complexity and feasibility of this problem.

A. *Problem Formulation*

To clearly present the problem we want to solve, we first formulate the TRP as a mathematic model. We note that the model presented below is different from that in [8] that does not consider varying topologies at different time instances. As we have discussed above, we should maintain the topology connectivity to guarantee the performance of delay sensitive flows, and then minimize the traffic loss during the reconfiguration. Accordingly, we minimize the total traffic loss under the topology connectivity constraint in the model. If f_{sd} and r^i_{sd} denote the required traffic rate and real traffic rate from ToR s to ToR d, respectively, the objective of the model should then be to minimize

$$\sum_{s,d}\sum_{i=1}^{n}(f_{sd}-r^i_{sd}) \tag{1}$$

under the topology connectivity constraints that can be summarized as

$$\boldsymbol{G}_{i-1}\cap\boldsymbol{G}_i \text{ is connected} \tag{2}$$

for all $1 \le i \le n$.

Equation (2) can also be explicitly expanded as

$$\sum_v c^{sdi}_{uv} - \sum_v c^{sdi}_{vu} \begin{cases} \ge 1, & \text{if } u = s \\ \le -1, & \text{if } u = d \\ = 0, & \text{otherwise} \end{cases} \tag{3}$$

for all $s \ne d$ and $1 \le i \le n$, where c^{sdi}_{uv} is used to indicate if link (u,v) is used to maintain the connectivity from ToR s to ToR d when topology $\boldsymbol{G}_{i-1}$ is converted to topology $\boldsymbol{G}_i$. This constraint implies that there is at least one path between any source-destination pair on the common part of two adjacent topologies, i.e., the topology connectivity is maintained during the configuration.

In addition, each topology should satisfy the constraints inherent in an OSA system, such as the maximum nodal degree constraint

$$\sum_v e^i_{uv} \le D \tag{4}$$

for all ToRs u, each link should be assigned special wavelengths across its end ToRs,

$$\sum_v W^{ki}_{uv} \le 1 \tag{5}$$

for all k, u and i, two ToRs should assign the same wavelengths to the link connecting them,

$$W^{ki}_{uv} = W^{ki}_{vu} \tag{6}$$

for all k, i and links (u,v), and the wavelength number constraint,

$$\sum_k\sum_v W^{ki}_{uv} \le W_N \tag{7}$$

for all u and i. In addition, a wavelength can only be assigned to a link that is on the common part of two adjacent topologies, and if a link is used to maintain the connectivity, there must be at least one wavelength assigned to it, i.e.,

$$W_{uv}^{ki} \le a_{uv}^{i} \tag{8}$$

and

$$c_{uv}^{sdi} \le W_{uv}^{ki} \tag{9}$$

for all i, k, s, d, u, v.

Furthermore, the flow related constraints should also be adhered too. That is

$$\sum_{s,d} l_{uv}^{sdi} \le C \sum_{k} W_{uv}^{ki} \tag{10}$$

for all links (u, v) and i, and

$$r_{sd}^{i} \le f_{sd} \tag{11}$$

for all links (u, v) and i, and

$$\sum_{v} l_{uv}^{sdi} - \sum_{v} l_{vu}^{sdi} = \begin{cases} r_{sd}^{i}, & \text{if } u = s \\ -r_{sd}^{i}, & \text{if } u = d \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

for all i and u. The left term in Constraint (10) is the traffic rate on link (u, v), which cannot exceed the total bandwidth that can be provided by all the wavelengths on link (u, v). Constraint (11) limits the real traffic rate between a pair of ToRs to not exceed their demands, while Constraint (12) is the flow conservation that must be satisfied in all the network traffic problems.

We note that there is an inherent relationship between the variables e_{uv}^{i} and a_{uv}^{i}, i.e., a_{uv}^{i} is 1 if and only if both e_{uv}^{i} and e_{uv}^{i-1} are 1. This can be formulated as

$$\frac{e_{uv}^{i-1} + e_{uv}^{i} - 1}{2} \le a_{uv}^{i} \le \frac{e_{uv}^{i-1} + e_{uv}^{i}}{2} \tag{13}$$

for all u, v, i.

In summary, the problem we will solve can be formulated as the following TRP:

minimize (1)

subject to: (3)–(13).

B. Feasibility and Complexity Analysis

Though we have formulated the TRP problem as a mathematic model, it cannot be easily solved in a reasonable amount of time, as it is not a convex programming problem. Though we can convert TRP to a mixed integer linear programming problem by fixing the number of intermediate topologies, it cannot be solved even for a small sized network (with less than 6 ToRs, 3 nodal degree, and 5 intermediate topologies). We can see that the TRP is NP-hard from the following theorem.

Theorem 1. The TRP is NP-hard.

Proof. We note that there are three subproblems in TRP, namely design of a sequence of topologies to guarantee the network connectivity, wavelength assignment and traffic routing. It is well known that wavelength assignment is an edge coloring problem that is NP-hard. Furthermore, the traffic routing to maximize the network throughput (or minimize the used capacity) is a multi-commodity flow problem, which is also NP-hard. Therefore, the TRP must be an NP-hard problem. □

Since TRP is NP-hard, we study if the TRP is always feasible and how to derive a feasible solution. With such a feasible solution, we can at least guarantee the performance of delay sensitive flows in DCNs. It should be noted that if we can find a set of variable values to satisfy (3)–(9) and (13), constraints (10)–(12) can always hold by setting r_{sd}^{i} to be a small number, i.e., the network carries very little traffic on the common part of two adjacent topologies. Accordingly, there are two key points on the feasibility of the TRP. First, there is a sequence of topologies that can be formed by the OSA such that: 1) the first topology is the initial topology; 2) the last topology is the target topology; 3) the common part of any two adjacent topologies is connected. Second, there is a wavelength assignment scheme to guarantee that at least one wavelength is assigned to the links used to maintain the network connectivity.

Since the topology formed by OSA is a D-regular graph, any nodes on the common part of two adjacent topologies have nodal degree at most D. Hence, once there are $D + 1$ wavelengths available, there is a wavelength assignment to guarantee that every link in the common part of two adjacent topologies has at least one wavelength. This must be satisfied in real OSA-based DCNs, since at least $D + 1$ wavelengths are required to fully utilize all the D-regular topologies. Accordingly, we only focus on the first key problem on the feasibility of TRP, i.e., the existence of intermediate topologies to maintain the network connectivity.

Theorem 2. If $\boldsymbol{G}$ and $\boldsymbol{G}^*$ are two connected 2-regular graphs with the same node set, there does not exist a graph sequence $\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_{n-1}$, such that

1) $\boldsymbol{G} \cap \boldsymbol{G}_1$, $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ for $i = 2, 3, \ldots, n-1$ and $\boldsymbol{G}_{n-1} \cap \boldsymbol{G}^*$ are connected graphs;

2) $\Delta(\boldsymbol{G}_i) \le 2$ for $i = 1, 2, \ldots, n-1$.

Proof. As connected 2-regular graphs, $\boldsymbol{G}$, $\boldsymbol{G}^*$ and all the intermediate topologies are cycles or lines. If $\boldsymbol{G}_i$ and $\boldsymbol{G}_{i+1}$ are two adjacent topologies, there are at least two links, say e_1 and e_2, such that $e_1, e_2 \in \boldsymbol{G}_i$ but $e_1, e_2 \notin \boldsymbol{G}_{i+1}$, since at least two links need to be adjusted for changing the position of nodes on a cycle or line. Hereby, $e_1, e_2 \notin \boldsymbol{G}_i \cap \boldsymbol{G}_{i+1}$. By removing e_1, e_2 from $\boldsymbol{G}_i$, it is disjointed and so is $\boldsymbol{G}_i \cap \boldsymbol{G}_{i+1}$.

Theorem 3. If $\boldsymbol{G}$ and $\boldsymbol{G}^*$ are two connected k-regular graphs ($k \ge 3$) with the same node set, there must be a graph sequence $\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_{n-1}$, such that

1) $\boldsymbol{G} \cap \boldsymbol{G}_1$, $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ for $i = 2, 3, \ldots, n-1$ and $\boldsymbol{G}_{n-1} \cap \boldsymbol{G}^*$ are connected graphs;

2) $\Delta(\boldsymbol{G}_i) \le k$ for $i = 1, 2, \ldots, n-1$.

To prove this theorem, we first introduce several lemmas.

Lemma 1. For any connected k-regular graphs ($k \ge 3$) $\boldsymbol{G}$, there is a graph sequence $\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_{n-1}$, such that

1) $\boldsymbol{G} \cap \boldsymbol{G}_1, \boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ for $i = 2, 3, \ldots, n-1$ are connected graphs;

2) $\boldsymbol{G}_{n-1}$ is a cycle;

3) $\Delta(\boldsymbol{G}_i) \le k$ for $i = 1, 2, \ldots, n-1$.

Proof. We prove this lemma by constructing a graph sequence satisfying all the three items. The Algorithm 1 is

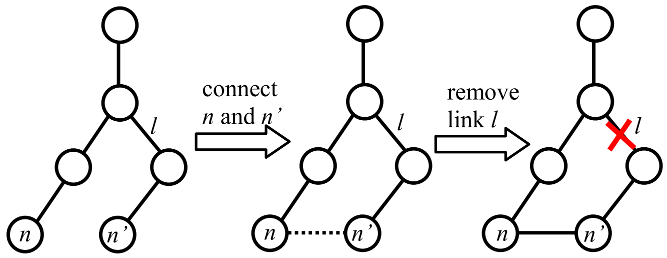


Fig. 3. Example for the "while" loop in Algorithm 1.

Algorithm 1 Transforming a graph to a cycle

```
Input: A connected graph $\boldsymbol{G}$
Output: A graph sequence $\{\boldsymbol{G}_i\}$ satisfying Lemma 1
1: $i \leftarrow 1$
2: Calculate a spanning tree of $\boldsymbol{G}$ and set it to be $\boldsymbol{G}_i$
3: while $\boldsymbol{G}_i$ has more than 2 leaves do
4:     Select 2 leaves in $\boldsymbol{G}_i$, say $n$ and $n'$.
5:     $\boldsymbol{G}_{i+1} \leftarrow \boldsymbol{G}_i + (n, n')$, $i \leftarrow i+1$
6:     Start at $n'$ to find out the first edge connecting a
       node with more than 2 nodal degree, say it is $l$.
7:     $\boldsymbol{G}_{i+1} \leftarrow \boldsymbol{G}_i - l$, $i \leftarrow i+1$
8: end while
9: Say $n$ and $n'$ is the last two leaves, $\boldsymbol{G}_{i+1} \leftarrow \boldsymbol{G}_i + (n, n')$
10: return $\{\boldsymbol{G}_i\}$
```

designed to construct such graph sequence. Next, we prove that the graph sequence generated by Algorithm 1 is satisfying all the three items in Lemma 1.

Item 1: We note that $\boldsymbol{G} \cap \boldsymbol{G}_1 = \boldsymbol{G}_1$ is connected since $\boldsymbol{G}_1$ is the spanning tree of $\boldsymbol{G}$. For odd i, $\boldsymbol{G}_{i+1}$ is derived by adding an edge on $\boldsymbol{G}_i$, and hence we have $\boldsymbol{G}_{i+1} \cap \boldsymbol{G}_i = \boldsymbol{G}_i$. That means $\boldsymbol{G}_{i+1} \cap \boldsymbol{G}_i$ is connected if and only if $\boldsymbol{G}_i$ is connected. For even i, since $\boldsymbol{G}_{i+1}$ is derived by deleting an edge from $\boldsymbol{G}$, we have $\boldsymbol{G}_i \cap \boldsymbol{G}_{i+1} = \boldsymbol{G}_{i+1}$. Note that, $\boldsymbol{G}_i$ is derived from connecting two leaves, say node n and n', on $\boldsymbol{G}_{i-1}$, there must be a cycle in $\boldsymbol{G}_i$ and node n and n' are on this cycle. If we start traversing from node n or n' in $\boldsymbol{G}_i$, and assume l is the first edge connecting a node with nodal degree larger than 2, removing link l from $\boldsymbol{G}_i$ does not divide $\boldsymbol{G}_i$ into multiple disconnected partitions, i.e., $\boldsymbol{G}_i \cap \boldsymbol{G}_{i+1} = \boldsymbol{G}_{i+1}$ is still connected. Accordingly, we know $\boldsymbol{G}_i \cap \boldsymbol{G}_{i+1}$ is a connected graph for all i. It is worth noting that $\boldsymbol{G}_i$ is always a tree for odd i, since there is only $N-1$ edges in the connected graph, where N is the number of nodes in $\boldsymbol{G}_i$.

Item 2: Fig. 3 shows how the "while" loops in Algorithm 1 (Line 3–8) works. In each loop, two leaves in the graph are eliminated and at most one leaf is generated. After $N-2$ loops, there are at most two leaves and the graph must be a line. In this case, a cycle can be formed by connecting these two leaves.

Item 3: Since $\boldsymbol{G}$ is a k-regular graph and $\boldsymbol{G}_1$ is the spanning tree of $\boldsymbol{G}$, there must be $\Delta(\boldsymbol{G}_1) \leq k$. For odd i, $\boldsymbol{G}_{i+1}$ is derived from connecting two leaves which can only generate two nodes with nodal degree $2 < k$. For even i, $\boldsymbol{G}_{i+1}$ is yielded by deleting an edge from $\boldsymbol{G}_i$, which cannot increase the nodal degree. □

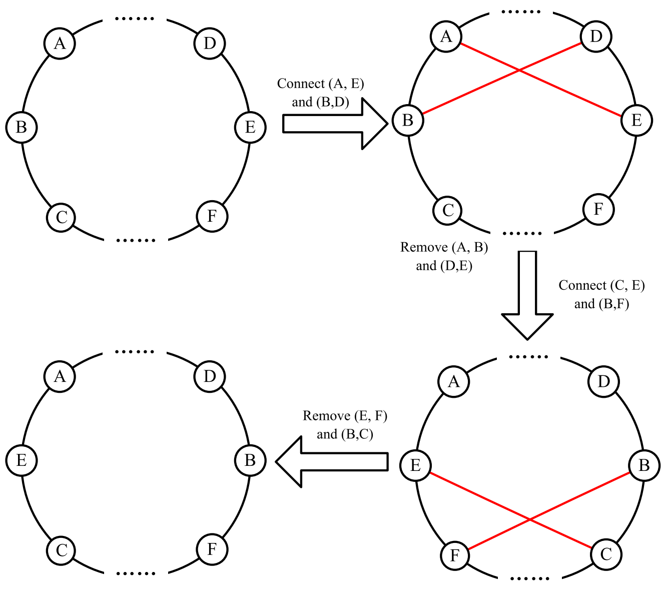


Fig. 4. How to swap the location of two nodes on the cycle.

Lemma 2. For two cycles, $\boldsymbol{C}$ and $\boldsymbol{C}^*$, that have the same node set and there are at least 4 nodes in each cycle , there must be a graph sequence, $\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_{n-1}$, such that

1) $\boldsymbol{C} \cap \boldsymbol{G}_1$, $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ for $i = 2, 3, \ldots, n-1$ and $\boldsymbol{G}_{n-1} \cap \boldsymbol{C}^*$ are connected graphs;

2) $\Delta(\boldsymbol{G}_i) \leq 3$ for $i = 1, 2, \ldots, n$.

Proof. To prove this lemma, it is enough to show that there is a graph sequence that satisfies the two items in the lemma to swap any two nodes on the cycle. As shown in Fig. 4, we want to swap the location of node B and node E. We first connect (A, E) and (B, D). Now, removing edge (A, B) and (D, E) does not impact the graph connectivity, and the node order on the upper side of the cycle becomes the desired order. After executing the same procedure on the lower side of the cycle as in Fig. 4, node B and E are swapped without changing the location of other nodes. In particular, if there are less than 2 nodes between the two nodes to be swapped, only one intermediate topology is required since we only need to fix the node order on one side of the cycle.

Based on the proof of Lemma 2, we can simply design Algorithm 2 to convert one cycle to another. It is worth noting that all the transformation procedures in both Algorithm 1 and Algorithm 2 are reversible, i.e., if the graph sequence $\{\boldsymbol{G}_i\}$ can transform $\boldsymbol{G}$ to a cycle $\boldsymbol{C}$, the inverse sequence of $\{\boldsymbol{G}_i\}$ can transform the cycle $\boldsymbol{C}$ to $\boldsymbol{G}$, so is the procedure to convert a cycle to another. Accordingly, the Theorem 3 can be proved by constructing such graph sequence. Hereafter, we use $\{\boldsymbol{G}_i\}^{-1}$ to denote the inverse sequence of $\{\boldsymbol{G}_i\}$.

Proof. Assume graph sequence $\{\boldsymbol{A}_i\}$ can transform $\boldsymbol{G}$ to a cycle $\boldsymbol{C}$ and $\{\boldsymbol{B}_i\}$ can transform $\boldsymbol{G}^*$ to a cycle $\boldsymbol{C}^*$. Both sequences can be obtained by Algorithm 1. $\{\boldsymbol{M}_i\}$ can convert $\boldsymbol{C}$ to $\boldsymbol{C}^*$, which can be derived by Algorithm 2. In this case, the graph sequence $\{\{\boldsymbol{A}_i\}, \{\boldsymbol{M}_i\}, \{\boldsymbol{B}_i\}^{-1}\}$ which satisfies all the items in Theorem 3 can convert $\boldsymbol{G}$ to $\boldsymbol{G}^*$ without impacting the network connectivity.

Algorithm 2 Convert one cycle to another

```
Input: Initial cycle $C$, final cycle $C^*$
Output: a graph sequence $\{G_i\}$ satisfying Lemma 2
1: Label the location of nodes with the same method (e.g.,
   label each node with $1, 2, \ldots$ clockwise) in both $C$ and
   $C^*$
2: $i \leftarrow 1$
3: while $G_i$ is not the same as $C^*$ do
4:     if $C(i) \neq C^*(i)$ then
5:         //$C(i)$ denotes the node at the location $i$ in $C$
6:         Swap $C(i)$ and $C^*(i)$ in $C$ and get all the
           intermediate graphs $G_i$
7:     end if
8: $i \leftarrow i + 1$
9: end while
10: return $\{G_i\}$
```

In Lemma 2, we restrict the maximum nodal degree to be 3. If we relax this constraint a little, we will get a much easier mean to fix the cycle.

Proposition 1. For two cycle graphs, C and C^*, that have the same node set and at least four nodes, there must be a graph G', such that

1) $C \cap G'$ and $G' \cap C^*$ are connected graphs;

2) $\Delta(G') \leq 4$.

Proof. The equivalent proposition of Proposition 1 is that for a cycle (N nodes) with unordered unique labels $(1, 2, \ldots, N)$ on each nodes, there must be a graph G' satisfying the two items in Proposition 1, such that the node labels in C^* are ordered. G' can be constructed by connecting node m with node $(m+1)\mathrm{mod}N$ and node $(m+N-1)\mathrm{mod}N$ if they are not connected in C. Clearly, there is a cycle in G' with ordered node labels and the nodal degree of each node is at most 4.

This proposition greatly reduces the number of intermediate topologies that is needed in Lemma 2.

IV. Algorithm Design

As we have discussed in the last section, the TRP is NP-hard and hence, it is difficult to find an optimal solution in a timely manner. Therefore, an efficient heuristic is necessary to solve this problem. To this end, we propose the TMA to solve the TRP described in Section IV-A. It is proved that TRP always has feasible solutions. Therefore, TMA should ensure that it can at least derive a feasible solution, and then further reduce the traffic loss during topology reconfiguration. In addition, we also discuss how to implement the TMA in real DCNs in Section IV-B.

A. Topology Management Algorithm

Based on the proof of Theorem 3, we can propose an algorithm to derive a sequence of intermediate topologies that can guarantee network connectivity during the topology reconfiguration. We note that reconfiguring the topology directly following the proof of Theorem 3 is not the ideal way in terms of minimizing the traffic loss during reconfiguration. For example, we first convert any graph to a cycle in the proof of Theorem 3. In fact, in addition to the cycle we used in the proof, many other links can still be maintained to carry traffic. Accordingly, the algorithm that directly follows the proof of Theorem 3 may incur too much traffic loss during topology reconfiguration, and we need some further improvements to reduce the traffic loss.

We adopt two major mechanisms to minimize the traffic loss during the topology configuration. The first one is to reduce the topology reconfiguration duration, i.e., minimize the number of intermediate topologies. To achieve this, some of the intermediate topologies in Theorem 3 are merged or eliminated.

- In Algorithm 1, some of the "while" loops can be merged. In each loop, we can connect leaves pair by pair until we are left with at most two leaves in the graph. When removing edges, we can sequentially determine which edge to remove for each pair of leaves. This is done to generate the most number of nodes with nodal degree 2 in each loop, so that we can use the least intermediate topologies to form a cycle that contains all the nodes in the graph.
- We note that in Algorithm 1, $G_{i-1} \cap G_{i+1} = G_i$ for all odd i, i.e., G_i is the common part of G_{i-1} and G_{i+1} for all odd i. Since G_i is connected, we can eliminate all intermediate topologies with odd subscript.
- In Algorithm 2, some of the node pairs can also be concurrently swapped. It should be noted that in each swapping procedure shown in Fig. 4, we only add a few links to swap the location of two nodes. In fact, we can add more links in the cycle to fix the location of more nodes with one intermediate topology. Therefore, we can try to swap the location of more than one pair of nodes simultaneously, if only the swapping process does not violate the nodal degree constraint.

In addition, we also try and fully utilize the network resources during the topology reconfiguration by using the following:

- When the spanning tree is calculated (Line 2 in Algorithm 1), the edge carrying more one-hop traffic volume is maintained with higher priority so as to minimize the total traffic in the network.
- Without increasing the number of intermediate topologies, we maintain as many links on the intermediate topologies as possible. This is to provide more routing choices for the traffic, and reduce the traffic loss during the topology reconfiguration.
- On every intermediate topology, wavelengths should also be adjusted according to the topology and traffic routed to serve as much traffic as possible. For example, the wavelengths on the link that will be deleted can be assigned to the links that will be maintained to serve more traffic.

Based on the above discussions, we propose the TMA to control the topology reconfiguration in OSA-based DCNs. TMA follows the idea to prove Theorem 3 that can ensure a feasible solution can be derived and also takes the above discussion into account to reduce the traffic loss during topology reconfiguration. This algorithm is shown in Algorithm 3.

Line 3–11 are used to calculate the intermediate topology sequence to guarantee the network connectivity during the topology reconfiguration based on the proof of Theorem 3. Different

Algorithm 3 Topology Management Algorithm

```
Input: Initial topology $\boldsymbol{G}$, final topology $\boldsymbol{G}^*$, traffic matrix
    $\{f_{ij}\}$, maximum nodal degree $D$
Output: Intermediate network management schemes
    $\{(\boldsymbol{G}_i, \boldsymbol{R}_i, \boldsymbol{W}_i)\}$
 1: Initialize $i \leftarrow 1$, $\boldsymbol{G}_0 \leftarrow \boldsymbol{G}$
 2: Set weight to every edge $(i, j)$ on $\boldsymbol{G}$ as $f_{ij}$
 3: Calculate maximum spanning tree of $\boldsymbol{G}$ which is $\boldsymbol{G}_i$
 4: while $\boldsymbol{G}_i$ has more than 2 leaves do
 5:     Make leaf pairs to connect and get intermediate
        topology $\boldsymbol{G}_i$
 6:     Remove edges as in Algorithm 1 and get $\boldsymbol{G}_i$,
        $i \leftarrow i + 1$
 7: end while
 8: Connect remaining two leaves, calculate routing and
    wavelength assignment
 9: Repeat line 4 to 8 on $\boldsymbol{G}^*$ and network management
    schemes are $\{\boldsymbol{G}^*\}$
10: Convert the cycle derived from $\boldsymbol{G}$ to the one derived
    from $\boldsymbol{G}^*$, say the intermediate topologies are $\{\boldsymbol{G}'\}$
11: $\{\boldsymbol{G}_i\} \leftarrow \{\boldsymbol{G}_i\} + \{\boldsymbol{G}'\} + \{\boldsymbol{G}^*\}^{-1}$
12: for $i$ from 1 to $n$ do
13:     for all link $l \in \boldsymbol{G}_{i-1}$ do
14:         if $\Delta(\boldsymbol{G}_i \cup l) \leq D$ then
15:             $\boldsymbol{G}_i \leftarrow \boldsymbol{G}_i \cup l$
16:         end if
17:     end for
18:     Sort all $(i, j)$ pairs, such that link $l = (i, j)$ is not
        on $\boldsymbol{G}_{i-1} \cup \boldsymbol{G}_i$ in decreasing order of $f_{ij}$; let $\boldsymbol{L}'$ be
        the set of such node pairs.
19:     for all links $l \in \boldsymbol{L}'$ do
20:         if add $l$ in $\boldsymbol{G}_i$ does not void degree constraint
            then
21:             $\boldsymbol{G}_i \leftarrow \boldsymbol{G}_i \cup l$
22:         end if
23:     end for
24:     Route all traffic on $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ by shortest path
and
        get $\boldsymbol{R}_i$
25:     Assign wavelength to $\boldsymbol{G}_{i-1} \cap \boldsymbol{G}_i$ by edge coloring
        and get $\boldsymbol{W}_i$
26: end for
27: return $\{(\boldsymbol{G}_i, \boldsymbol{R}_i, \boldsymbol{W}_i)\}$
```

from the proof of Theorem 3, some intermediate topologies have been merged in TMA to minimize the configuration duration. For example, TMA only reserves the intermediate topologies with even subscripts in Algorithm 1, and takes all the leaves to make pairs and connect in each "while" loop from Line 4 to 8.

Once the topology sequence is derived, TMA adds more links into the intermediate topologies to provide higher connectivity in order to reduce the traffic loss during the reconfiguration (Line 13–23). For $\boldsymbol{G}_i$, the links on $\boldsymbol{G}_{i-1}$ have higher priority to be added (Line 13–17), as they can be used to carry traffic when the topology is transformed from $\boldsymbol{G}_{i-1}$ to $\boldsymbol{G}_i$. After that, other links can also be added if the nodal degree constraint is satisfied (Line 18–23). Now, the links that can carry more direct traffic (i.e., with larger f_{ij}) have higher priority to be added. This is to minimize the used capacity, and hence reduce the traffic loss in the network.

In addition to guaranteeing the topology connectivity, TMA also calculates the intermediate routing (Line 24) and wavelength assignment (Line 25). When routing the traffic in the intermediate topologies, TMA uses the shortest path scheme as it can minimize the total used capacity. For the wavelength assignment, TMA uses multi-graph edge coloring algorithm to serve as much traffic as possible on the intermediate topologies. With this algorithm, if there are $D + P$ wavelengths available in the system (i.e., every ToR supports $D + P$ wavelengths), where D is the maximum nodal degree and P is the maximum number of wavelengths required on each link, all the traffic can be served. It is worth noting that both the routing and wavelength assignment are limited to the common part of the two adjacent topologies, since only the common part is unchanged during the topology transformation.

When we transform the spanning tree to a cycle, each "while" loop in Algorithm 3 will eliminate $\lfloor \frac{L}{2} \rfloor$ leaves on the graph, where L is the number of leaves on the graph $\boldsymbol{G}_i$. Even if the spanning tree is a star-type network (i.e., has the most number of leaves), at most $\lceil \log(N) \rceil$ intermediate topologies are required to convert a spanning tree to a cycle, where N is the number of ToRs in the network. When we transform a cycle to another, at most $N - 2$ intermediate topologies are needed. Accordingly, there are $2\lceil \log(N) \rceil + N - 2$ intermediate topologies to complete the network reconfiguration in the worst case.

B. Discussion

To implement TMA in real DCNs, there remain some challenges. First, we cannot change the MEMS connection, wavelength assignment and traffic routing simultaneously, as this scheme cannot guarantee that a link is removed only when all the traffic on it is rerouted to another path. This is because the MEMS configuration may be changed before all the data transmissions on the removed link(s) are complete. To prevent this, we first update the traffic routing, and then change the wavelength assignment, and only then reconfigure the MEMS in the final step. In this way, the network connectivity can be actually maintained during the reconfiguration.

Second, TMA is implemented on the DCN central controller. When the computation completes, all the reconfiguration procedure can be pre-installed into each component in the network and then enabled simultaneously. This method requires an accurate global clock in the DCNs. This can be realized by technologies such as [14] that can achieve microsecond-level synchronization in DCNs without additional hardware.

Third, TMA ignores the time to reconfigure the WSS, which can also cause some traffic loss. However, we note that this reconfiguration time is only about 10 μs by the state-of-the-art technology such as [15]. This time is negligible even for the mice flows, since the inter-ToR RTT in a DCN is usually above 400 μs [12].

Fourth, the topology reconfiguration may cause TCP packet disordering. However, it may not be an emergent problem in TMA. One reason is that TMA does not allow a flow to send packets along multiple paths concurrently, and hence there are few out of order packets. The other reason is that in high throughput low latency DCNs, the TCP sending window will recover very soon when packet disordering occurs. Thus, the reconfiguration does not cause much performance degradation. If no packet disordering can be tolerated then we can hold each flow for 200 μs to eliminate the TCP packet disordering, since 200 μs is larger than half of the RTT in DCNs. On the other hand, as 200 μs is usually less than the time interval between two flowlets, i.e., the bursts of packets in a flow that are separated by large gaps [16], it will not impact the flow performance.

In addition, though TMA can ensure that there is a connected topology during each step of the topology reconfiguration, it cannot guarantee a traffic-lossless reconfiguration. To guarantee the performance of delay sensitive flows, we can enable mice flows to preempt other traffic in the network by assigning them a high priority. In this case, the traffic overflow only occurs on the elephant flows, and usually a temporary traffic overflow is not too much overhead for elephant flows.

We note that most of the intermediate topologies are used to convert a cycle to another in our algorithm. To reduce the number of intermediate topologies and the traffic overflow, we suggest constructing an OSA-based network with maximum nodal degree of at least 4, which is very practical in OSA-based DCNs. In this case, we need at most $2\lceil\log(N)\rceil + 1$ intermediate topologies to complete the topology reconfiguration. That is $\lceil\log(N)\rceil$ intermediate topologies to transform initial topology to a cycle, $\lceil\log(N)\rceil$ intermediate topologies to transform a cycle to the final topology, and one intermediate topology to transform one cycle to another. In addition, if the algorithm to calculate the optimal topology for each time instance inherently creates a cycle containing all the ToRs, such as the algorithm proposed in [8], we can directly use this cycle instead of the one found by Algorithm 1 to reconfigure the topology. In this case, only 2 intermediate topologies are required to complete the reconfiguration process, which is exactly the case in our simulation.

V. Evaluation

In this section, we evaluate the performance of the TMA through extensive simulation experiments. In Section V-A, we study the performance of TMA in reducing the traffic loss during the topology reconfiguration. Since TMA has to perform in real-time in an online system, the algorithm running time is an important issue. This is investigated in Section V-B. We also discuss the system cost incurred by our TMA, e.g., the configuration duration and the impact of TCP disordering in Section V-C.

All the traffic traces in our simulation were collected from a production data center with 10 778 VMs hosted by IBM global services. Fig. 5 shows the characteristics of the traffic in our data set. Fig. 5(a) shows that the total traffic in the network changes frequently from instance to instance, and hence dynamically

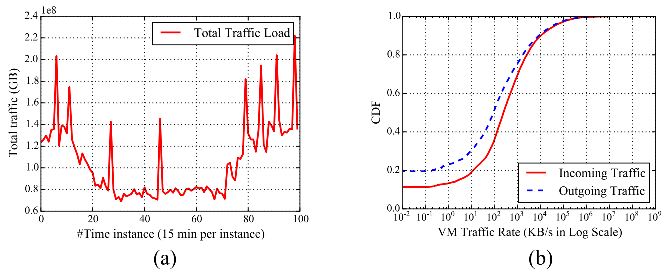


Fig. 5. Traffic statistic. (a) Total traffic variance. (b) The CDF of traffic rate.

reconfiguring the network topology is necessary to reduce the used network capacity and serve more traffic. Fig. 5(b) shows the cumulative distribution function (CDF) of the incoming and outgoing traffic rate of VMs in the network. The figure indicates that there are lots of mice flows in the network, and hence we need TMA to guarantee the network connectivity.

The dataset only provides the incoming and outgoing traffic rates of each VM, whereas we need traffic matrix to calculate the optimal topology at each time instance. In addition, the traffic matrix is also a part of the input of TMA. To solve this problem, we apply the Gravity model [17] to estimate the traffic matrix. To show the performance of delay sensitive flows with and without TMA, we assume that 10% of the traffic belongs to delay sensitive flows, and will be assigned a higher priority. The optimal topology in each time instance is calculated by the algorithm introduced in [8].

A. *TMA Performance*

In this section, we study the performance of TMA in OSA-based DCNs with different OSA parameters, i.e., the available wavelengths and the maximum nodal degree. In the production data center, we collect the traffic information every 15 min and then update the network topology. For a 25-h data collection, there are 99 time instances in these figures. In all the experiments, we assume there are 80 ToRs and all the VMs are evenly allocated to these ToRs (i.e., there are $\lfloor\frac{10778}{80}\rfloor = 134$ or $\lfloor\frac{10778}{80}\rfloor + 1 = 135$ VMs to each ToR). We study how to progressively complete the topology reconfiguration in OSA-based DCNs, and compare traffic loss in the cases that are with and without progressive topology reconfiguration.

1) Impact of Available Number of Wavelengths: Fig. 6 shows the performance of TMA in the networks with different number of available wavelengths. To study the impact of number of available wavelengths, we fix the maximum nodal degree of each ToR to 4 (the same as in [8]). From these figures, we can make three main observations. First, TMA can always prevent the loss of mice flows irrespective of the number of available wavelengths in the network. This is the most important benefit brought by TMA as it maintains the network connectivity during the topology reconfiguration and assigns a high priority to the mice flows.

Second, regardless of the number of available wavelengths, the traffic loss (i.e., both mice flows and elephant flows) during topology reconfiguration is identical without TMA. The reason for this is that the traffic routing is independent of the number

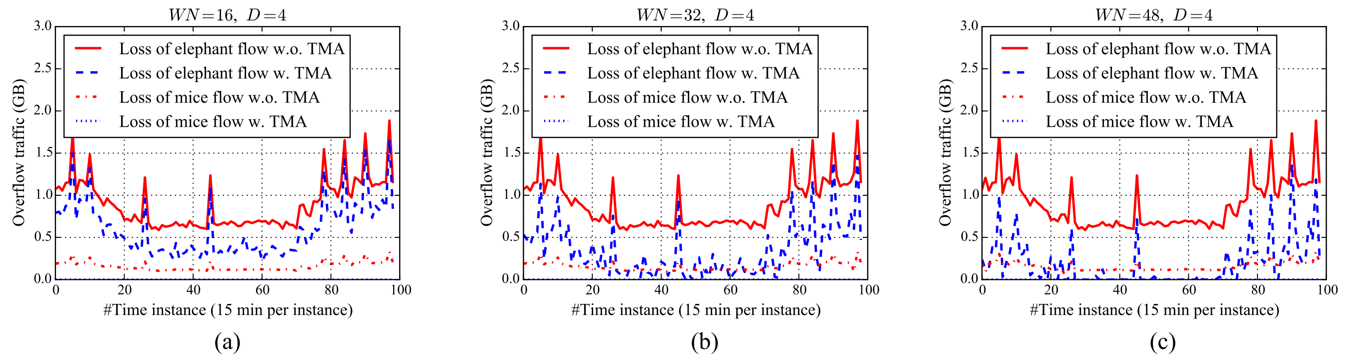


Fig. 6. TMA performance with different number of wavelengths. (a) 16 available wavelengths. (b) 32 available wavelengths. (c) 48 available wavelengths.

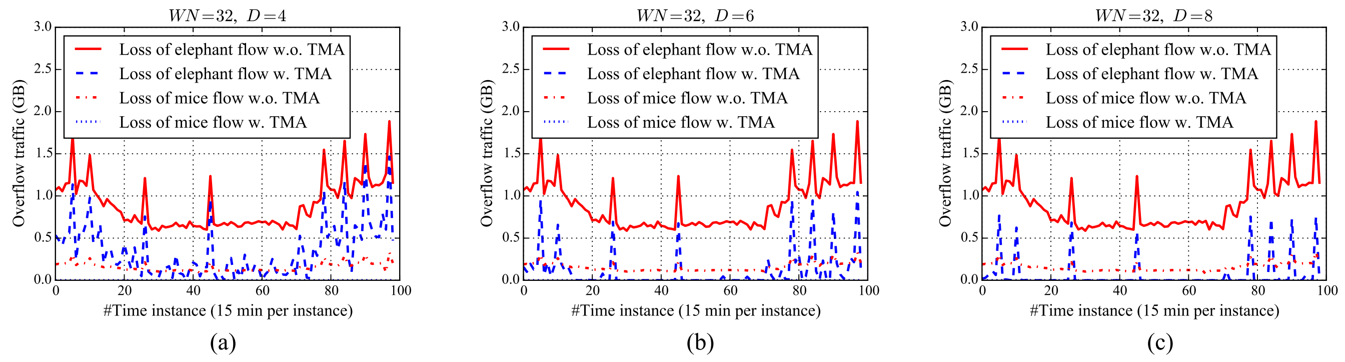


Fig. 7. TMA performance with different maximum nodal degree. (a) Maximum nodal degree = 4. (b) Maximum nodal degree = 6. (c) Maximum nodal degree = 8.

of wavelengths in the network. The amount of lost traffic is only determined by the removed number of links during the topology reconfiguration.

The last observation is that TMA can always reduce the traffic loss, and the more the number of available wavelengths, the higher the reduction in traffic loss by TMA. When there are only 16 wavelengths in the network, TMA can reduce about half of the traffic loss in most cases. When there are 48 wavelengths available, TMA can realize traffic lossless reconfiguration in most of the time instances. This is because that TMA adjusts the wavelength configuration, i.e., change the link capacity, to serve more traffic during the topology reconfiguration. With more wavelengths in the network, there is more bandwidth that can be used to carry the traffic and hence a reduction in the traffic loss during the reconfiguration.

2) Impact of Maximum Nodal Degree: Another parameter that may impact the performance of TMA is the maximum nodal degree of each ToR in the network. To investigate how this parameter impacts the TMA performance, we fix the number of available wavelengths to be 32 and vary the maximum nodal degree. The simulation results are shown in Fig. 7.

From this figure, we can make three observations. The first observation is that the TMA can always maintain the transmission of mice flows regardless of the maximum nodal degree. This is again due to the network connectivity maintained by TMA, and hence we can guarantee the transmission of mice flows by assigning them a high priority.

In addition, we observe that larger nodal degree cannot bring significant improvement for an OSA-based DCN without using TMA in terms of the traffic loss during topology reconfiguration. On the one hand, larger nodal degree can reduce the traffic in the network and provide the potential to reduce traffic loss during reconfiguration. On the other hand, more links should be adjusted in the network with larger nodal degree resulting in larger traffic loss when we convert a topology to another, and no matter which scheme is adopted we should adjust more links resulting in more traffic loss. Thus, we note that larger nodal degree cannot reduce traffic loss significantly without using TMA.

Third, we can see that TMA performs better in the networks with larger maximum nodal degree. In a network with certain number of ToRs, larger nodal degree means more available links and OSA can form more topologies, and hence traffic has more routing choices during topology reconfiguration. This provides larger optimization space to TMA and TMA can further reduce the traffic loss by leveraging such larger optimization space.

These simulations show that compared with reconfiguring the topology in one step, TMA cannot only maintain network connectivity and protect the performance of delay sensitive flows, but also always reduce the traffic loss during the topology reconfiguration in OSA-based DCNs. More network resources, such as more wavelengths, larger maximum nodal degree (i.e., more links), can provide TMA larger optimization space and TMA can further reduce traffic loss by fully utilizing these resources.

B. Time Complexity

In TMA, the main factor that impacts the algorithm running time is the network size, i.e., number of nodes and links.

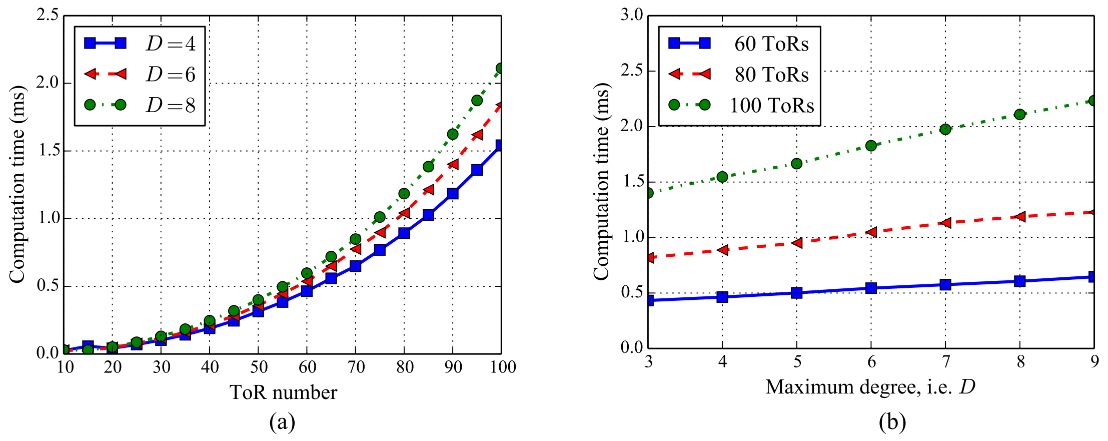


Fig. 8. Algorithm running time. (a) Network size versus algorithm running time. (b) Maximal node degree versus algorithm running time.

Accordingly, we investigate how the number of ToRs in the network (which impacts the number of nodes) and the maximum nodal degree (which impacts the number of links) affects the algorithm running time. All algorithm running times are calculated on a laptop with Dual-Core 2.70 GHz AMD CPU. The simulation results are shown in Fig. 8.

From Fig. 8(a), we can see that though the running time of TMA will increase with the network size, it is still relatively small compared with the time to reconfigure MEMS. When there are 100 ToRs in the network, with maximum nodal degree of each ToR, TMA requires only 1.5 ms to calculate the progressive reconfiguration method (including intermediate topologies, traffic routing and wavelength assignment).

From Fig. 8(b), we observe that the algorithm running time also increases with the increase in the maximum nodal degree of each ToR. However, this increase is gradual and very slowly. For example, as the maximum nodal degree changes from 3 to 9, TMA only requires 42.35% more time to calculate the progressive reconfiguration approach though there are three times more edges in the network.

C. System Cost

The TMA uses intermediate topologies and routing to bridge various topology configurations in OSA. However, it may bring about two main costs to the OSA system. One is an increase in the reconfiguration time due to the use of intermediate topologies, while the other is TCP packet disordering incurred due to the routing changes during topology reconfiguration.

We adopt the algorithm in [8] to calculate the optimal topology configuration. The topology derived by this algorithm ensures that there is always a cycle containing all the nodes in the topology. As we discussed in Section IV-B. we can directly use this cycle instead of the one derived by Algorithm 1 in TMA. Hence, only two intermediate topologies are required, and the reconfiguration duration is 30 ms.

Compared with the one-step reconfiguration method, TMA may result in more blocking of flows in the last 20 ms during a topology reconfiguration, as fewer network resources can be used due to the progressive reconfiguration procedure. However, it can save more traffic during the first 10 ms, since TMA utilizes as many network resources as possible to keep the network connectivity and serve more traffic. According to our simulation results, we can see that TMA can save more traffic than one-step reconfiguration method. Therefore, a slightly larger traffic blocking probability is cost efficient for the network operator. More importantly, TMA maintains the network connectivity and provides better performance to the delay sensitive flows.

To investigate the impact of TCP packet disordering incurred by changing flow route(s) in TMA, we use NS3 [18] to do a packet level simulation. Different from previous simulations, we only inject one flow into the network and change the topology and flow routing according to a topology reconfiguration sequence obtained in previous simulation. We only do a limited simulation in NS3 as doing a full-fledged packet level simulation (i.e., with a large number of flows, packets, network size) is too time consuming. In the simulation, we use TCP New Reno to control the TCP window size, set the retransmission timeouts to be 10 ms as in data center TCP [13], and assume that the round-trip time (RTT) is 100 μs between the VMs under two directly connected ToRs. In addition, we assume that each additional hop switching adds a delay of 11 μs to a packet, which consists of one unit of processing delay (about 9.8 μs [19]) and one unit of transmission delay (1.2 μs in a 10 Gb/s network). The buffer size at each ToR port is set to 125 kB so as to fully utilize the network bandwidth without packet disordering.

Under the above settings, we study how the TCP transmission rate changes with time during the topology reconfiguration. For a fair comparison, we reconfigure the network topology every 30 ms. Fig. 9 shows the simulation result. From Fig. 9(a), we can see that TCP packet get disordered with TMA. However, packets of a flow are never sent concurrently through multiple paths, and hence, the TCP window size will recover very soon when the TCP window reduces due to the TCP packet disordering. Without TMA, the traffic will be held for at least 10 ms, which may incur a timeout to the TCP connection and reduces the sending window size to 1. Thus, TMA benefits TCP flows and helps in maintaining a high throughput.

Fig. 9(b) shows how the traffic rate of a TCP flow changes during the topology reconfiguration. From this figure, we see that

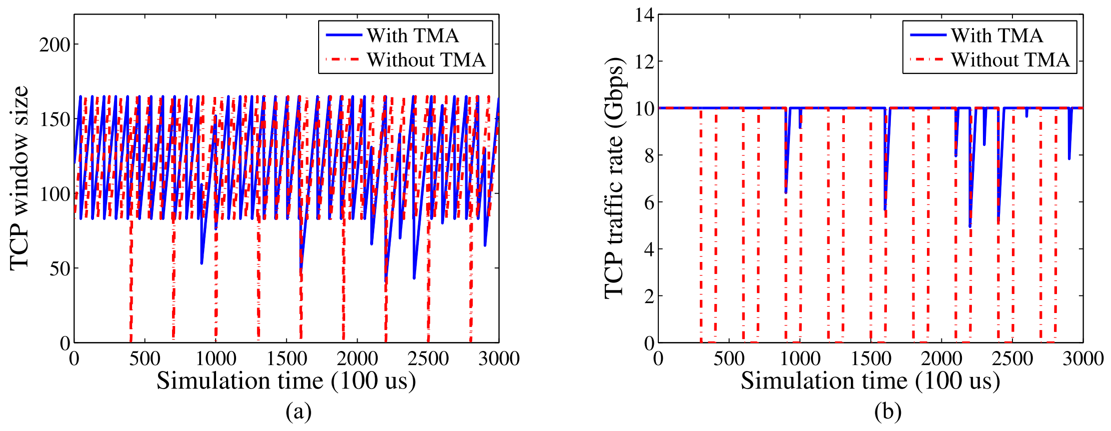


Fig. 9. Impact of TCP packet disordering. (a) How TCP window changes. (b) How traffic rate changes.

the TCP packet disordering only impacts the traffic throughput slightly and that too temporarily, since the TCP sending window will recover soon even if a TCP packet disordering occurs. However, without TMA, the flow will be blocked for at least 10 ms, which greatly reduces the network throughput. Thus, TMA reduces the traffic loss during topology reconfiguration.

VI. Related Work

In this work we design an intermediate plan to update different network states (such as routing, topology, link weight etc.) to guarantee the performance of delay sensitive flows and reduce the total traffic loss. There are some existing works that focus on providing guarantees during network updates [20], [21], 22], [23]. SWAN [20] dynamically change traffic routing to fully utilize network resource. In order to prevent traffic loss during the routing reconfiguration, SWAN reserves some redundant capacity on each link and uses a series of linear programming models to calculate a routing modification plan, such that no data loss is incurred by asynchronous routing update. However, SWAN only focuses on the traffic routing changes but not on the topology changes as in our work. zUpdate [21] also focuses on intermediate routing scheme, but it works on different scenario. zUpdate is proposed for the case that some switches will be shut down to upgrade or some VM migration will occur. In this case, the network topology or the traffic matrix will be changed. zUpdate gives out a network update scheme to complete these update without traffic congestion in the network. Similar to SWAN, zUpdate only changes the traffic routing but not the topology as in our work.

In addition, R-BGP [22] pays attention to failures, i.e., links go up or down in the Internet, when the dynamics of BGP may cause packet loss. R-BGP pre-computes a few strategically chosen failover paths, and provably guarantees that a domain will not become disconnected. Compared with R-BGP that directly changes traffic routing, [23] proposed a methodology to seamless modifying the configuration of commonly used link-state IGP. By determining the link state changing order, [23] guarantees that the migration will not create IP transit service outages. However, none of these works design intermediate topologies for performance guarantee. To the best of our knowledge, TMA is the first algorithm to provide guarantees by leveraging not only temporary routing, but also intermediate topologies.

VII. Conclusion

To handle the ever increasing traffic in DCNs in a scalable manner we need technologies that can dynamically and rapidly change the network topology. In this work we have studied how to manage the topology reconfiguration in OSA-based DCNs. To keep the network connectivity to guarantee the performance of delay sensitive flows and minimize the traffic loss during the topology reconfiguration, we leverage intermediate topologies (including the routing and wavelength assignment on these topologies) to realize the reconfiguration progressively. To get these intermediate topologies, we first formulated this TRP as a mathematic model, and then analyzed the complexity and feasibility of the problem. From the analysis, we find that the TRP is NP-hard, but we can always find a feasible solution to guarantee the network connectivity during the reconfiguration. Accordingly, we propose a heuristic TMA that can at least find such feasible solution, and then reduce the traffic loss during the network topology reconfiguration. Through extensive simulation experiments we show that the TMA algorithm can reduce the traffic loss during the topology reconfiguration to a large extent with very short algorithm running time.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.
[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “Vl2: A scalable and flexible data center network,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009.
[3] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, “Bcube: A high performance, server-centric network architecture for modular data centers,” in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.
[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] A. Hamza, J. Deogun, and D. Alexander, “Free space optical data center architecture design with fully connected racks,” in *Proc. IEEE Global Telecommun. Conf.*, 2012, pp. 2192–2197.

[6] J. P. Srikanth Kandula and P. Bahl, "Flyways to de-congest data center networks," in *Proc. 8th ACM Workshop Hot Topics Netw.*, 2009, pp. 1–6.
[7] W. Zhang, X. Zhou, L. Yang, Z. Zhang, B. Y. Zhao, and H. Zheng, "3d beamforming for wireless data centers," in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, pp. 1–6.
[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," presented at the Symp. Networked Systems Design Implementation, San Jose, CA, USA, 2012.
[9] 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," in *Proc. SIGCOMM*, 2010, pp. 339–350.
[10] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *Proc. SIGCOMM*, 2010, pp. 327–338.
[11] 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.*, New York, NY, USA, 2010, pp. 267–280.
[12] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. (2009, Aug.). Safe and effective fine-grained TCP retransmissions for datacenter communication. *SIGCOMM Comput. Commun. Rev.* 39(4). pp. 303–314. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592604
[13] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan,"Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, New York, NY, USA, 2010, pp. 63–74.
[14] E. Mallada, X. Meng, M. Hack, L. Zhang, and A. Tang, "Skewless network clock synchronization without discontinuity: Convergence and performance," *IEEE/ACM Trans. Netw.*, 2014, to be published.
[15] N. Farrington, A. Forencich, P.-C. Sun, S. Fainman, J. Ford, A. Vahdat, G. Porter, and G. C. Papen, "A $10\mu s$ hybrid optical-circuit/electrical-packet network for datacenters," presented at the Optical Fiber Communication Conf., Anaheim, CA, USA, 2013.
[16] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM SIGCOMM*, 2014, pp. 503–514.
[17] Y. Zhang, M. Roughan, N. Duffield, and A. Greeberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *Proc. ACM SIGMETRICS*, 2003, pp. 206–217.
[18] Ns3. (2015). [Online]. Available: https://www.nsnam.org/.
[19] K. Mahmood, A. Chilwan, O. N. Sterb, and M. Jarschel. On the modeling of openflow-based SDNs: The single node case. (2015). [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1411/1411.4733.pdf
[20] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 15–26.
[21] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate: Updating data center networks with zero loss," in *Proc. ACM SIGCOMM*, New York, NY, USA, 2013, pp. 411–422.
[22] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: Staying connected in a connected world," presented at the 4th USENIX Conf. Networked Systems Design Implementation, Cambridge, MA, USA, 2007.
[23] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Seamless network-wide IGP migrations," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 314–325.

Yangming Zhao received the B.S. degree in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in July 2008, where he is currently working toward the Ph.D. degree. His research interests include network optimization and data center networks.

Sheng Wang (M'09) received the B.S. degree in electronic engineering and the M.S. and Ph.D. degrees in communication engineering all from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in July 1992, 1995, and 2000, respectively. He is a Professor at UESTC. He is a Senior Member of the Communication Society of China, a Member of the ACM, and a Member of the China Computer Federation. His research interests include planning and optimization of wire and wireless networks, next generation of Internet, and next-generation optical networks.

Shouxi Luo is currently working toward the Ph.D. degree with the University of Electronic Science and Technology of China, Chengdu, China. His research interests include data center networks and software-defined networking.

Vishal Anand received the B.S. degree in computer science and engineering from the University of Madras, Chennai, India, in 1996, and the M.S. and Ph.D. degrees in computer science and engineering from the University at Buffalo, The State University of New York (SUNY), Buffalo, NY, USA, in 1999 and 2003, respectively. He was a Research Scientist at Bell Labs, Lucent Technologies and Telcordia Technologies (ex-Bellcore). He is currently an Associate Professor at the College at Brockport, SUNY. His research interests include the area of wired and wireless computer communication networks including optical networks and cloud and grid computing. He received the "Rising Star" and the "Promising Inventor Award" Award from the Research Foundation of SUNY, and the "Visionary Innovator" Award from the University of Buffalo.

Hongfang Yu received the B.S. degree in electrical engineering from Xidian University, Xian, China, in 1996, and the M.S. and Ph.D. degrees in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2006, respectively. From 2009 to 2010, she was a Visiting Scholar at the Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA. Her research interests include network survivability and next-generation Internet, cloud computing, etc.

Xiong Wang received the B.S. degree in electronic and information engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, and the Ph.D. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2003 and 2008, respectively. Since then, he has been with the School of Communication and Information Engineering, University of Electronic Science and Technology of China, where he is currently an Associate Professor.

Shizhong Xu received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1994, 1997 and 2000, respectively. He is currently a Professor at UESTC. His research interests include broadband networks, all-optical network, and next-generation networks.

Xiaoning Zhang received the B.S., M.S., and Ph.D. degrees in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2002, 2005, and 2007, respectively. He is currently an Associate Professor at the Key Lab of Broadband Optical Fiber Transmission and Communication Networks, School of Communication and Information Engineering, University of Electronic Science and Technology of China. His research interests include network design and optical and broadband networks.