photonics

MDPI

Article

CASM: A Cost-Aware Switch Migration Strategy for Elastic Optical Inter-Datacenter Networks

Yong Liu [1], Qian Meng [2], Zhonghua Shen [2] and Fulong Yan [3,*]

[1] School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China; yongliu@hznu.edu.cn
[2] School of Mathematics, Hangzhou Normal University, Hangzhou 311121, China; mq@hznu.edu.cn (Q.M.); szh@hznu.edu.cn (Z.S.)
[3] Alibaba Cloud, Alibaba Group, Beijing 100102, China
* Correspondence: yanfulong.yfl@alibaba-inc.com

Abstract: In inter-datacenter elastic optical networks, multi-controller deployment is adopted to improve the stability and scalability of the control plane. As the network scale increases, the traditional multi-controller deployment scheme ignores the dynamic characteristics of traffic, resulting in unbalanced load among multiple controllers. In response to this problem, the existing switch migration mechanism is proposed to achieve balanced distribution of control loads. However, most of the existing research work does not consider the additional cost of switch migration, and the load balancing performance of the controller is not significantly improved after switch migration. In this paper, we propose a cost-aware switch migration (CASM) strategy for controller load balancing. The proposed CASM strategy first measures the controller load through multiple performance indicators that affect the controller load, and then judges whether the controller is overloaded or underloaded based on the controller's response time to the request message, thereby improving the load balancing performance of the controller. Additionally, when selecting the switch to be migrated, the CASM selects the optimal switch for migration based on minimizing the migration cost, thereby reducing the cost of switch migration. The performance evaluation shows that CASM significantly improves load balancing performance of controllers and reduces the migration cost compared to existing solutions.

Keywords: elastic optical networks; controller deployment; switch migration; load balancing; migration cost

Citation: Liu, Y.; Meng, Q.; Shen, Z.; Yan, F. CASM: A Cost-Aware Switch Migration Strategy for Elastic Optical Inter-Datacenter Networks. *Photonics* **2022**, *9*, 315. https://doi.org/10.3390/photonics9050315

Received: 27 February 2022
Accepted: 4 May 2022
Published: 6 May 2022

1. Introduction

With the expansion of network scale and the sudden growth of traffic, communication among data centers (DCs) [1,2] requires high network performance to support fast data transmission, data backup, and data synchronization. However, the traditional inter-DC networks, in which electrical switching is the core technology, encounter technical bottlenecks in terms of bandwidth capacity, energy consumption, and transmission latency. To address the high performance requirements of inter-DC communication, elastic optical networks (EONs) [3–8] are widely regard as the most promising technology for DC interconnecting. EONs can provide a relatively better infrastructure for meeting the high-bandwidth and low-latency requirements of interconnecting datacenters. In the elastic optical inter-datacenter network, in order to effectively control the state information and network resources of the whole network, the software-defined network (SDN) technology is proposed [9,10]. SDN is a network virtualization technology, which realizes centralized control of the entire network by separating the control plane and the forwarding plane [11,12]. For multi-tenant service requests, the SDN controller can adaptively allocate network resources, improving the flexibility of the network [13–15].

In the actual network operation process, a single SDN controller can easily handle the flow request messages generated by the small-scale network without overloading.

However, when the network scale expands and the flow request messages grow in bursts, the SDN controller will be overloaded due to the processing of a large number of request messages. In response to this problem, multi-controller deployment is proposed, which has achieved distributed management and control by dividing the network into multiple domain networks. In Figure 1a, three SDN controllers control domain networks of different scales, respectively. Due to the different scales of each domain network, the processing overhead of each controller for the flow request message is different, so this will cause load imbalance among multiple controllers. In order to solve the above problems, existing work proposes a dynamic switch migration strategy for load balancing among multiple controllers. As shown in Figure 1b, controller load balancing can be achieved by migrating switches among the domain networks associated with the three controllers.

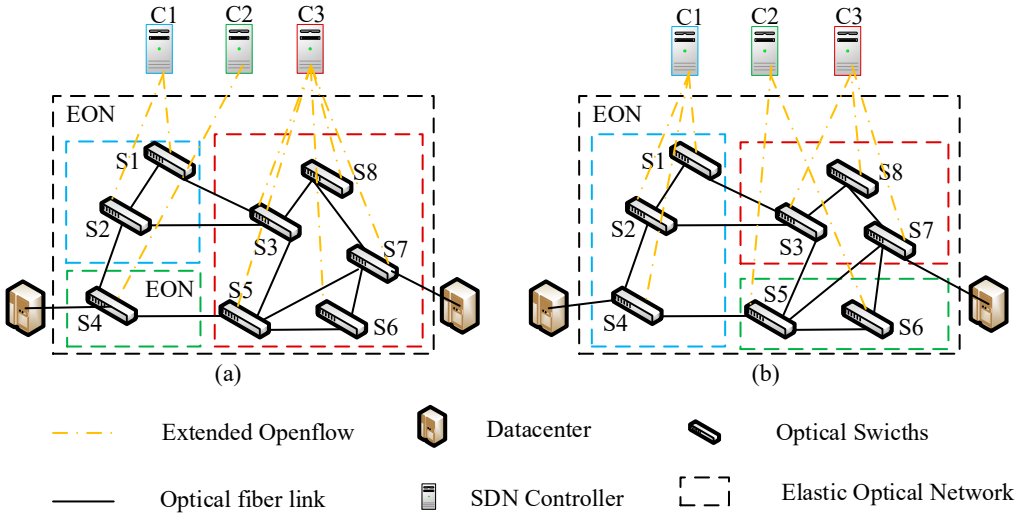

Figure 1. An example of the multi-controller deployment: (**a**) controller load imbalance; (**b**) controller load balancing.

In the case of dynamic traffic changes, through the switch migration optimization mechanism, the balanced distribution of loads among multiple controllers can be effectively achieved, thereby improving the stability of the control plane. During actual network execution, the switch migration mechanism selects the switch to be migrated from the set of switches controlled by the overloaded controller and migrates it to the set of switches controlled by the underloaded controller. In the existing research work, most of them use the amount of request messages sent by the switch as a measure of the controller load, and then use the difference in the load of different controllers to judge whether the controller is overloaded. In addition, they select the switch with the highest flow request rate from the domain network controlled by the overloaded controller and migrate it to the domain network controlled by the underloaded controller to complete the switch migration activity. However, the challenge faced by most existing switch migration mechanisms is that they ignore other performance indicators that affect the load of the controller, such as the calculation and formulation overhead of routing rules by the controller, which leads to low load balancing performance of the controller. At the same time, when selecting switches to be migrated, they ignore the additional cost of switch migration, which leads to higher migration costs.

In response to the above problems, in this work, we design and propose a cost-aware switch migration (CASM) strategy to achieve balanced distribution of controller load and improve the effectiveness of switch migration. The main contributions of this work are as follows:

- Different from the existing research work that takes the amount of flow request messages sent by the switch as the controller load indicator, our proposed CASM strategy takes the controller's processing overhead of flow request messages and rule

formulation as the controller load indicator at the same time. Additionally, based on the load of different controllers, the CASM strategy accurately measures whether the controller is overloaded by calculating the average response time of the controller, thereby improving the load balancing performance of the controller;

- Most of the existing research work usually selects the switch with the highest flow request rate as the switch to be migrated, ignoring the migration cost of the switch. In response to this problem, the proposed CASM strategy reduces the migration cost of switches in the switch migration activities by defining multiple performance indicators that affect the migration cost, and selecting the optimal switch based on the minimum migration cost;
- Experimental simulations show that the proposed CASM strategy improves the load balance performance of the controller by 43.2% and reduces the switch migration cost of the by 41.5% compared with existing research schemes [16–18].

The remainder of this paper is organized as follows. In Section 2, we discuss related works. In Section 3, the detailed design of CASM strategy is presented. In Section 4, we evaluate the performance of CASM through Mininet simulations. Finally, we conclude the paper in Section 5.

2. Related Works

Aiming at the uneven distribution of multi-controller loads, Dixit et al. [16] achieved switch migration activities by selecting switches from the domain network controlled by the overloaded controller and associating them with the controllers adjacent to the overloaded controller. In the actual network execution process, since it does not consider whether the target controller is overloaded, after switch migration, this migration strategy may cause the target controller to be overloaded, resulting in poor load balancing performance of the controller. In order to improve the load balancing performance of the controller, Chen et al. [19] proposed a switch migration mechanism based on the zero-sum game theory, which associates the selected switch to be migrated with the light-loaded controller. However, this kind of migration mechanism needs to select multiple switches to be migrated and light-loaded controllers in one migration activity. Since the load of the controllers changes dynamically in each time period, after the switches are migrated, the relationship between the controllers and switches may not be optimal. To improve the effectiveness of switch migration, Yu et al. [20] proposed a load notification-based switch migration mechanism. In the actual execution process, when the controller is overloaded, the system will notify the controller that the overload occurs, and the migration mechanism is activated, and the distribution of the controller load has been adjusted by migrating switches between domain networks controlled by multiple controllers. To achieve load balancing among multiple controllers, Cello et al. [21] proposed a heuristic-based switch migration optimization mechanism (BalCon). The mechanism uses a heuristic algorithm to select the best matching relationship between the controller and the switch. In order to improve the effectiveness of switch migration, Lan et al. [22] proposed a switch migration mechanism based on a game decision-making mechanism. This mechanism selects the optimal target controller to associate with the switch to be migrated by maximizing the resource utilization of the controller, thereby improving the load balancing performance of the controller. In order to effectively measure whether the controller is overloaded, Cui et al. [23] used the average response time of the controller to flow request messages to determine whether the controller is overloaded. If the average response time of the controller is greater than a given threshold, the controller is identified as overloaded, otherwise it is underloaded. Through the analysis of the above solutions, we can see that most of the existing solutions may cause the target controller to be overloaded after the switch is migrated, resulting in unsatisfactory load balancing performance of the controller. In addition, during the switch migration process, the switch migration will generate additional migration costs, resulting in large energy consumption of the network.

During the switch migration process, the response time of the controller to the flow request message is also an effective indicator to measure the load performance of the

controller. In order to improve the migration efficiency of switches, Zhou et al. [24] selected a group of switches from the set of switches controlled by the overloaded controller as the switches to be migrated, and implemented multiple switches to migrate in one switch migration activity, thereby improving the load balancing of the controller. However, during the switch migration process, this solution does not consider the switch migration cost. Cui et al. [23] judged whether the controller is overloaded or underloaded by the average response time of the controller to the flow request message, thereby improving the load balancing performance of the controller. However, during the switch migration process, this mechanism also does not consider the switch migration cost. During the switch migration process, Sahoo et al. [25] used the Karush-Kuhn-Tucker condition to find the best target controller, and then associated it with the switch to be migrated, thereby improving the switch migration efficiency. For the analysis of the above solutions, although the existing switch migration optimization mechanism can improve the migration efficiency of the switch by constraining the response time of the controller, the additional migration cost is not reduced after the switch is migrated, and the load balancing performance of the controller is not significantly improved.

In order to reduce the extra cost of switch migration activities, during the switch migration process, Wang et al. [26] selected the optimal switch from the domain network controlled by the overload controller as the switch to be migrated based on minimizing the migration cost, and migrated it to the domain network controlled by the target control. In order to improve the migration efficiency of switches, Hu et al. [27] calculated the controller load by considering multiple performance indicators that affect the controller load, and selected the optimal switch based on the minimized cost, so as to achieve a balanced distribution of the controller load. In addition, Hu et al. [28]. propose an efficient switch migration mechanism, which selects the switches to be migrated from the switch set by sensing the migration cost of the switches and associates them with light-load controllers. Through the above analysis, although the existing solution can reduce the cost of switch migration, there are other costs affecting switch migration that are not considered, and the load balancing performance of the controller is not significantly improved after switch migration.

From the analysis of the above research work, the existing switch migration strategy cannot make a good trade-off between the controller load balancing performance and the migration cost. For most research works, they only consider the flow request rate as a controller load metric. During the switch migration process, since there are multiple performance indicators that affect the load of the controller, this will result in a low load balancing performance of the controller. In addition, they incur significant migration costs during switch migration. In addition, for some solutions considering switch migration cost, they ignore some cost indicators that affect switch migration, and only consider one cost indicators, which will lead to the inability to significantly reduce switch migration cost. In this paper, the proposed CASM strategy calculates the load of the controller through multiple indicators, and accurately measures whether the controller is overloaded based on the controller's response time to flow request messages, which can achieve a balanced distribution of the controller load. In addition, the switch with the smallest migration cost is selected from the switch set associated with the overloaded controller based on multiple cost indicators that affect switch migration.

3. Detailed Design of CASM Strategy

To achieve balanced distribution of controller load, we propose a cost-aware switch migration (CASM) mechanism. As shown in Figure 2, we design the system framework of the CASM scheme. Based on SDN technology, the CASM system framework is divided into a control plane and a data forwarding plane, and a communication interface protocol (Extended OpenFlow) is defined between the control plane and the forwarding plane. On the data forwarding plane, multiple data centers are interconnected through elastic optical networks. In actual deployment, we use OpenFlow-enabled optical cross-connects

(OF-OXCs) as a node facility, and on each OF-OXCs node, OpenFlow protocol agent (OFP Agent) is deployed to receive messages sent by the controller. In the actual execution process, through the extended OpenFlow protocol [29–32], the controller centrally controls the node facilities of the forwarding plane. In the control plane, multiple controllers are deployed for distributed control of the entire network. Each controller is deployed with multiple functional modules, including the proposed CASM module.

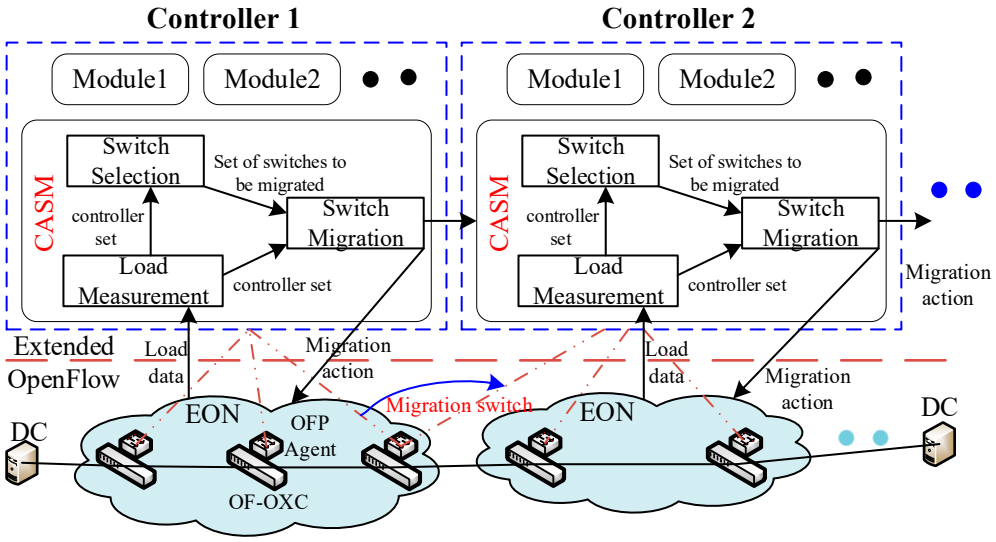

Figure 2. CASM system structure.

As the core functional module of the controller, the CASM module is mainly composed of three functions: (1) *Load Measurement* is used to identify whether the controller is overloaded, and output the overloaded controller set and the underloaded controller set; (2) *Switch Selection* is used to select the optimal switch from the set of switches controlled by the overloaded controller as the switch to be migrated; (3) *Switch Migration* is used to associate the switch to be migrated with the target controller to complete the switch migration activity. In the network execution process, the CASM strategy first collects the load data of the domain network associated with the SDN controller, and then identifies whether the controller is overloaded based on the load information of each controller, and outputs the overloaded controller set and the underloaded controller set, respectively. After that, the CASM strategy selects the switch with the least migration cost from the set of switches associated with the overload controller through the switch selection component and passes it to the switch migration component for dynamic switch migration activities. Finally, the CASM strategy delivers the generated switch migration function to the data forwarding plane, so as to implement switch migration activities among multiple domain networks.

The specific execution flow of the CASM scheme is shown in Figure 3. CASM first calculates the average response time of each controller, and judges whether the controller is overloaded based on the response time threshold τ. If the response time of the controller exceeds the threshold τ, the set ϑ of overloaded controllers is output; otherwise, the set ψ of underloaded controllers is output. In the actual execution process, the overloaded controller C_j^{large} with the largest load is first selected from the overloaded controller set ϑ for the switch migration activity. For overloaded controller C_j^{large}, CASM selects the optimal set φ of switches from the set $\lambda_{C_j^{\text{large}}}$ of switches it controls. Additionally, the underloaded controller C_k^{small} with the least load is selected from the set ψ of underload controllers. Based on the underloaded controller C_k^{small}, CASM selects the optimal switch S_i^{mini} from the set φ of switches to be migrated. Finally, the switch S_i^{mini} is migrated from the domain network of the overloaded controller C_j^{large} to the domain network controlled by the underloaded controller C_k^{small} to complete the switch migration activity. In the actual

execution process, according to the execution flow of the CASM scheme, the execution of the scheme is stopped until the overloaded controller set ϑ is empty.

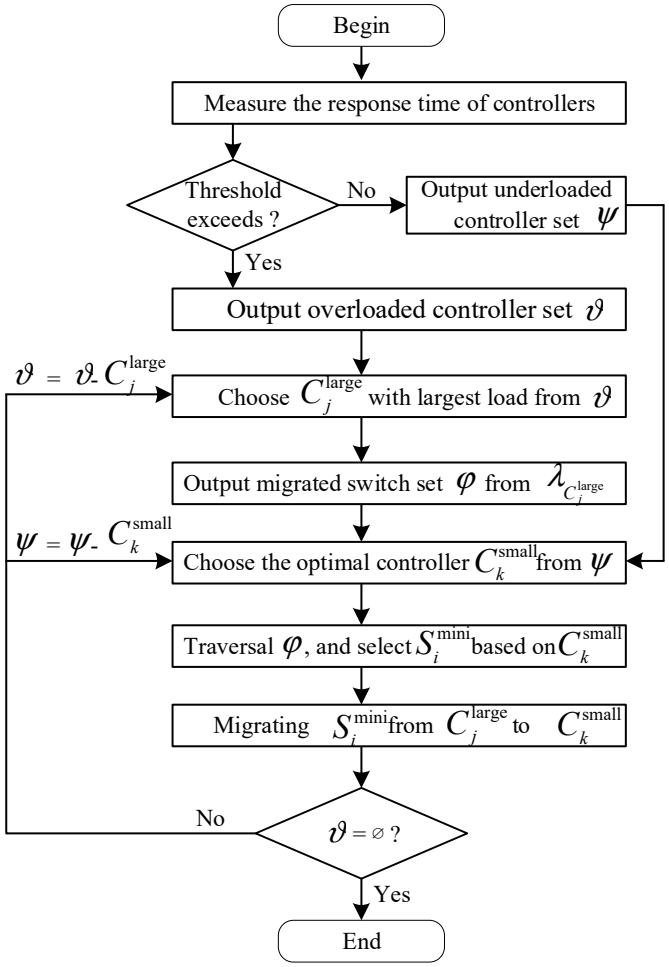

Figure 3. Flowchart of CASM framework.

In the actual execution process, we model the network model of the forwarding plane as a graph $G(V, E)$, where $V = S \cup C$ represents a series of node facilities, including switch and controller nodes. $E = \{e_1, e_2, \ldots, e_r\}$ represents a series of links. In the forwarding plane, $S = \{S_1, S_2, \ldots, S_m\}$ represents a series of switch nodes, where S_i refers to the ith $(i = 1, 2, \ldots, m)$ switch in the switch set S. Additionally, $C = \{C_1, C_2, \ldots, C_n\}$ represents a series of controller nodes, where C_j refers to the jth $(j = 1, 2, \ldots, n)$ controller in the controller set C. In actual operation, the network is divided into n sub-domain networks by deploying multiple controllers. $\lambda_{C_j} \in S$ represents the set of switches associated with the controller C_j. In Table 1, we list the main symbols and explain them. Below we mainly elaborate and explain the design of the proposed CASM scheme in detail.

Table 1. Notations in this study.

Notion	Meaning
S_i	The i-th switch
C_j	The j-th controller
λ_{C_j}	The set of switch associated with the controller C_j
m_{ij}	Static connection matrix within S_i and C_j
$P(S_i)$	The flow request message sent by the switch S_i
γ_{packet}	Average size of Packet_in message
δ_{rule}	Average size of rule formulation message
L_{C_j}	The load of controller C_j
t_{C_j}	Average response time of controller C_j
ϑ	A series of overloaded controller nodes
ψ	A series of underloaded controller nodes
φ	Switch set to be migration
S_i^{mini}	Switch S_i with minimal migration cost
C_j^{large}	Controller C_j with the largest load
C_k^{small}	Controller C_j with the least load
Φ_{S_i}	Migration cost of the switch S_i
S	A series of switch nodes
E	A series of links

3.1. Load Measurement

For the load measurement module, it is mainly used to calculate the load of different controllers, and at the same time judge whether the controller is overloaded or underloaded based on the response time of the controller to the request message. In the actual execution process, the performance indicators that affect the load of the controller mainly include two types: (1) the controller's processing overhead for flow request messages (Packet_in); (2) The controller's processing overhead for rule formulation messages (Packet_out). In actual execution, when a new flow arrives at the switch, it is forwarded by matching the flow table entry. If there is no matching flow table entry on the switch, the switch sends a Packet_in message to the controller. After the controller receives the flow request message, it will calculate the routing rules according to the state information of the whole network. After the routing rule is formulated, the controller sends the flow table entry to the switch with a packet_out message for flow matching and forwarding [33,34]. The matching relationship between the switch S_i and the controller C_j can be defined by a matrix $\mathrm{M} = \left[m_{ij}\right]_{m \times n}$. m_{ij} is a mapping relationship between the switch S_i and the controller C_j, as shown in Equation (1).

$$m_{ij} = \begin{cases} 1, S_i \in \lambda_{C_j} \\ 0, S_i \notin \lambda_{C_j} \end{cases}. \tag{1}$$

Let $P(S_i)$ denote the number of Packet_in messages sent by the switch S_i to controller C_j at time at a time interval T. γ_{packet} is the average size of a Packet_in message. The processing overhead L_j^1 of the controller C_j for flow request messages can be defined as Equation (2).

$$L_j^1 = \sum_{S_i \in \lambda_{C_j}} P(S_i) \times \gamma_{packet} \times m_{ij}. \tag{2}$$

Let δ_{rule} denote the average size of a rule formulation message (Packet_out). The processing overhead L_j^2 of the controller C_j for rule formulation messages (packet_out) is shown in Equation (3).

$$L_j^2 = \sum_{S_i \in S} \sum_{C_j \in C} P(S_i) \times \delta_{ruler} \times m_{ij}. \tag{3}$$

Therefore, to sum up the above, as shown in Equation (4), the load L_{C_j} of controller C_j is the sum of the above two performance metrics.

$$L_{C_j} = L_j^1 + L_j^2. \tag{4}$$

To measure whether the controller C_j is overloaded, we use the average response time t_{C_j} of the controller C_j to messages to identify. The messages mentioned here include flow request messages (Packet_in) and rule formulation messages (Packet_out). Let t_{in} denote the time when a Packet_in message arrive at the controller C_j. After the controller C_j processes the Packet_in message, a Packet_out message is generated and sent to the switch S_i. Let t_{out} denote the time when a Packet_out message arrive at the switch S_i. Therefore, we easily acquire the response time $t_{response}$ of the controller C_j to the single round-trip message between the switch S_i and the controller C_j, where the round-trip message refers to the Packet_in and Packet_out messages, as shown in the Equation (5).

$$t_{response} = t_{out} - t_{in}. \tag{5}$$

In addition, we denote the response time to the whole round-trip messages between the switch S_i and the controller C_j as $t_{S_i\_response}$. Therefore, the total response time of all the round-trip messages associated with the controller C_j can be easily obtained. After acquiring these data, we can directly obtain the average response time t_{C_j} of the controller C_j for a single message size, as shown in the Equation (6). Therefore, for all controllers $C_j \in C$, we can obtain the average response time of a single controller and use it as a threshold τ for the response time of the controller, as shown in the Equation (7).

$$t_{C_j} = \frac{\sum\limits_{S_i \in \lambda_{C_j}} t_{S_i\_response}}{L_{C_j}}. \tag{6}$$

$$\tau = \frac{\sum\limits_{C_j \in C} t_{C_j}}{n}. \tag{7}$$

The specific execution process of the load measurement module is shown in Algorithm 1. Algorithm 1 first calculates the loads L_{C_j} of all controllers $C_j \in C$, as shown in Equation (4). Then, based on the load of different controllers, the average response time t_{C_j} of the controller C_j can be obtained. Meanwhile, the algorithm use the set $\Lambda = \{t_{C_1}, t_{C_2}, \ldots, t_{C_n}\}$ represent the response time values of n controllers in a time interval. Finally, the algorithm traverses the value t_{C_j} in the set Λ. When $t_{C_j} > \tau$, the controller C_j is added to the overloaded controller set ϑ, otherwise, the controller C_j is added to the underloaded controller set ψ. Therefore, through this algorithm, the set ϑ of overloaded controllers and the set ψ of underloaded controllers can be output separately.

3.2. Switch Selection

In this module, switch selection is mainly used to select the best switch from the domain network associated with the overload controller as the switch to be migrated. In the actual implementation process, we choose the optimal switch by minimizing the migration cost. The number of switches associated with the overloaded controller $C_j \in C$ is defined as N_o, while the number of the underloaded controllers $C_k \in \psi$ is defined as N_u. In the switch selection process, there are two performance indicators that affect the switch migration cost, which are the change cost of the controller's processing overhead of the flow request message and the controller's deployment cost of the migration rule.

The change cost $\Phi_{i,k}^1$ of the controller's processing overhead of the flow request message is shown in Equation (8). In Equation (8), h_{ik} represents the number of hops from switch S_i to controller C_k, and h_{ij} represents the number of hops from switch S_i to controller C_j.

$$\Phi_{i,k}^{1} = P(S_i) \times \min(h_{ik}) - P(S_i) \times \min(h_{ij}) \times m_{ij}. \quad (8)$$

Algorithm 1: Load Measurement.

Input: S, C, $\mathrm{P}(S_i)$, γ_{packet}, δ_{rule}, m_{ij}, $t_{S_i\_response}$, τ

Output: ϑ, ψ

Initialization: Overloaded controller set ϑ is the empty set; Underloaded controller set ψ is the empty set
for $C_j \in C$ **do**
    Computer two load metrics (L_j^1, L_j^2), Get L_{C_j} for each controller C_j
    Obtain the average response time t_{C_j} of the controller C_j
end for
Get the set $\Lambda = \{t_{C_1}, t_{C_2}, ..., t_{C_n}\}$ to represent the response time value of n controller in a time interval
while Λ is not the empty set **do**
    if $t_{C_j} > \tau$ **then**
        Add C_j to the overloaded controller set ϑ
    else
        Add C_j to the underloaded controller set ψ
    end if
    $\Lambda = \Lambda$ - $\{t_{C_j}\}$
end while
return ϑ, ψ

In the actual execution process, the controller C_j formulates the corresponding migration rules and delivers them to the switch S_i, and the switch migration activity has been realized. Thus, the deployment cost $\Phi_{i,k}^{2}$ of the migration rule is shown in Equation (9). σ_{rule} represents the number of migration rule messages sent by controller C_j.

$$\Phi_{i,k}^{2} = \sigma_{rule} \times \min(h_{ij}) \times m_{ij}. \quad (9)$$

From the above analysis, we can see that the migration cost of the switch is actually composed of the change cost of the controller to the flow request message and the deployment cost of the controller to the migration rules. We modeled the two cost metrics separately above, so the migration cost Φ_{S_i} of the switch S_i can be expressed as Equation (10).

$$\Phi_{S_i} = \Phi_{i,k}^{1} + \Phi_{i,k}^{2}. \quad (10)$$

During the switch selection process, the switch selection module calculates the migration cost of all switches in the domain network associated with the overload controller $C_j \in \vartheta$, and selects the optimal switch S_i^{mini} as the switch to be migrated based on the minimum migration cost $\min(\Phi_{S_i})$. The specific execution process of switch selection is shown in Algorithm 2. The algorithm firstly calculates the two cost indexes that affect

switch migration, and then calculates the migration cost Φ_{S_i} of each switch $S_i \in \lambda_{C_j}$ based on this. Based on the migration costs Φ_{S_i} of different switches, the algorithm further constructs a migration cost set Ψ. Then, for the migration cost Φ_{S_i} of each switch in Ψ, the algorithm is based on minimizing the migration cost, and selects the switch S_i^{mini} with the smallest migration cost as the switch to be migrated by matching each underloaded controller $C_k \in \psi$. Finally, the algorithm forms all the selected switches into a switch set as the switch set φ to be migrated.

Algorithm 2: Switch Selection.

Input: $C_j \in \vartheta$, $C_k \in \psi$, $\mathrm{P}(S_i)$, λ_{C_j}, σ_{rule}

Output: Set of switches to be migrated: φ

1: **Initialization:** Number of switches associated with the overloaded controller C_j is N_o; Number of underloaded controller C_k is N_u
2: **for** $S_i \in \lambda_{C_j}$, $C_k \in \psi$ **do**
3: Computer two migration cost metrics $(\Phi_{i,k}^1, \Phi_{i,k}^2)$
4: Obtain the migration cost Φ_{S_i} of switches $S_i \in \lambda_{C_j}$ based on each underloaded controller $C_k \in \psi$
5: **end for**
6: Construct the migration cost set $\Psi = \{\Phi_{S_i} \mid 1 \le i \le \mathrm{N}_o, 1 \le k \le \mathrm{N}_u\}$ of switches $S_i \in \lambda_{C_j}$
7: **while** Ψ is not the empty set **do**
8: **for** $1 \le k \le \mathrm{N}_u$ **do**
9: For each underloaded controller C_k, select the switch S_i^{mini} from the set λ_{C_j} of switches based on minimizing the migration cost $\min(\Phi_{S_i})$
10: Add switch S_i^{mini} to the set φ of switches to be migrated
11: **end for**
12: $\Psi = \Psi - \{\min(\Phi_{S_i})\}$
13: **end while**
14: **return** φ

3.3. Switch Migration

For the switch migration module, the module selects the best switch S_i^{mini} from the set λ_{C_j} of switches associated with the overloaded controller $C_j \in \vartheta$ based on the minimum migration cost and associates it with the underloaded controller $C_k \in \psi$, thereby realizing the switch migration activity. In the existing research work, most research schemes can only handle one overloaded controller in one switch migration activity, but cannot handle multiple migration activities. Therefore, in our proposed work, in one switch migration activity, we are able to process multiple overloaded controllers in parallel by

multi-threading to improve the efficiency of switch migration. The specific execution process of switch migration is shown in Algorithm 3.

In actual execution, Algorithm 3 firstly measures whether the controller $C_j \in C$ is overloaded through the load measurement function module, and outputs the overloaded controller set ϑ and the underloaded controller set ψ at the same time. Then, the algorithm obtains the overloaded controller $C_j^{\text{large}} = \max_{C_j \in \vartheta}(L_{C_j})$ with the largest load from the set ϑ of overloaded controllers, and the underloaded controller $C_k^{\text{small}} = \min_{C_k \in \psi}(L_{C_k})$ with the least load from the set ψ of underloaded controllers. In addition, the algorithm calculates the switch S_i^{mini} with the smallest migration cost as the optimal switch from the set φ to be migrated. Finally, the algorithm migrates switches S_i^{mini} from the domain network associated with the overloaded controller C_j^{large} to the domain network associated with the underloaded controller C_k^{small}, thus completing the switch migration activity. In the actual running process, the proposed CASM scheme stops running until the overloaded controller set ϑ is empty.

Algorithm 3: Switch Migration.

Input: $S_{\text{i}} \in S$, $C_{\text{j}} \in C$, $\text{P}(S_i)$, λ_{C_j}, m_{ij}

Output: Migration actions set Θ

Initialization: Migration set Θ is the empty set
Calculate overloaded controller set ϑ and underloaded controller set ψ
while ϑ is not the empty set **do**
    $C_{\text{j}}^{\text{large}} = \max_{C_j \in \vartheta}(L_{C_j})$
    Obtain the optimal set φ of switches based on minimizing migration costs
    $C_{\text{k}}^{\text{small}} = \min_{C_k \in \psi}(L_{C_k})$
    for $S_{\text{i}} \in \varphi$ **do**
        Select the optimal switch $S_{\text{i}}^{\text{mini}}$ based on the controller $C_{\text{k}}^{\text{small}}$
    end for
    Add $\left\langle C_{\text{j}}^{\text{large}}, S_{\text{i}}^{\text{mini}}, C_k^{\text{small}} \right\rangle$ to migration set Θ
end while
return Θ

4. Performance Evaluation

In this section, in order to evaluate the performance of the CASM scheme, we leverage the Mininet simulation tool and Ryu controller to build a test platform. The experiment is run on a physical server which is a Supermicro X11DPL-i with two 12-core Intel Xeon Silver-4116 2.10 GHz CPUs and 64 GB memory, and an Intel Ethernet Connection X722 for 1 GbE. The server runs Ubuntu 16.04–64 bit system with Linux 4.15.0 kernel. In the actual testing process, we use topology structures of different network scales, which are all derived from actual industrial map models. We compare and analyze the performance of the proposed scheme and related research works under different topologies. The different topologies and features are shown in Table 2. These topologies all come from typical networking structures in the industry. For example, the NSFNET topology is a backbone network structure formed by connecting supercomputing centers in the United States.

Table 2. Characteristics of topologies.

Parameter / Topology	NSFNET [35]	OS3E [36]	Ta2 [37]	Interoute [38]
Number of nodes	14	34	65	110
Number of sides	21	42	108	159
Number of controllers	2	4	6	8

During the execution of the experiment, we used the Iperf software tool to simulate the typical traffic model in the data center network. Assume that there is a pair of bidirectional fibers on each link, and the available spectral width of each fiber is set to 4000 GHz with a width of 12.5 GHz. The bandwidth requirements of the links are randomly distributed between 12.5 Gb/s and 100 Gb/s during network execution. The flow requests are generated according to a Poisson process with a rate of r requests per service provision period, and the duration of a request follows an exponential distribution with an average value of v service provision periods. Hence, the traffic load can be quantified with $r \times v$ in Erlangs. To analyze the performance of the proposed scheme, we compare it with existing related research works, which include: **(1) Static Switch Matching (SSM):** it realizes the optimal deployment of the controller through the static association between the switch and the controller [17]; **(2) Random Switch Migration (RSM)**: it randomly selects the switch as the switch to be migrated during the switch migration activity [16]; **(3) Maximum Utilization Controller Selection (MUCS):** it selects the controller with the largest resources as the target controller during switch migration [18]. During the experiment, we choose two indicators that can reflect the performance of the scheme as the experimental evaluation indicators, which are the load imbalance degree of the controller and the migration cost of the switch. The load imbalance degree $\frac{\sum_{1\le j\le n}\left|L_{C_j}-L^*\right|}{n\times L^*}$ represents the load balancing performance among multiple controllers, where $L^* = \frac{\sum_{1\le j\le n} L_{C_j}}{n}$ represents the average load of the controller. From the analysis of performance data, the smaller its value is, the more it can reflect that the scheme has a better effect on the load balancing performance of the controller. On the contrary, the load balancing performance of the controller is low. Migration costs represent additional costs incurred in switch migration activities. When the migration cost index of the solution is too high, it indicates that the migration efficiency of the solution is low, resulting in high network energy consumption. If the migration cost index of the solution is small, it indicates that the solution is more efficient in switch migration activities.

During the actual experiment, we conduct experimental tests and analysis on the load imbalance indicators of different schemes, as shown in Figure 4. For the data values output by the experiment, we can find that the smaller the load imbalance index is, the better the load balancing performance of the controller is. Under different traffic loads, we can see that the load balancing performance of CASM is better than the other three schemes. Since in the switch migration activity, CASM uses different load indicators to calculate the load of the controller, and judges whether the controller is overloaded based on the response time of the controller to the request message, thereby improving the load balancing performance of the controller. In Figure 4a, we test the performance of the proposed CASM scheme based on the NSFNET topology. From the experimental data, it can be seen that compared with SSM, RSM and MUCS, the load balancing performance of the proposed CASM scheme is improved by 55.3%, 46.2% and 37.5%, respectively. As shown in Figure 4b, under the OS3E topology, compared with the existing scheme SSM, RSM and MUCS, the proposed CASM scheme improves the load balancing performance by 47.3%, 36.2% and 25.6%, respectively. Under the other two network topologies, compared with the existing research schemes, the proposed CASM scheme also has better load balancing performance. Since in the implementation process of the proposed CASM scheme, in addition to considering the performance indicators that affect the load of the controller, the response time of the

controller to the message is also considered, which makes the CASM scheme have better load balancing performance.

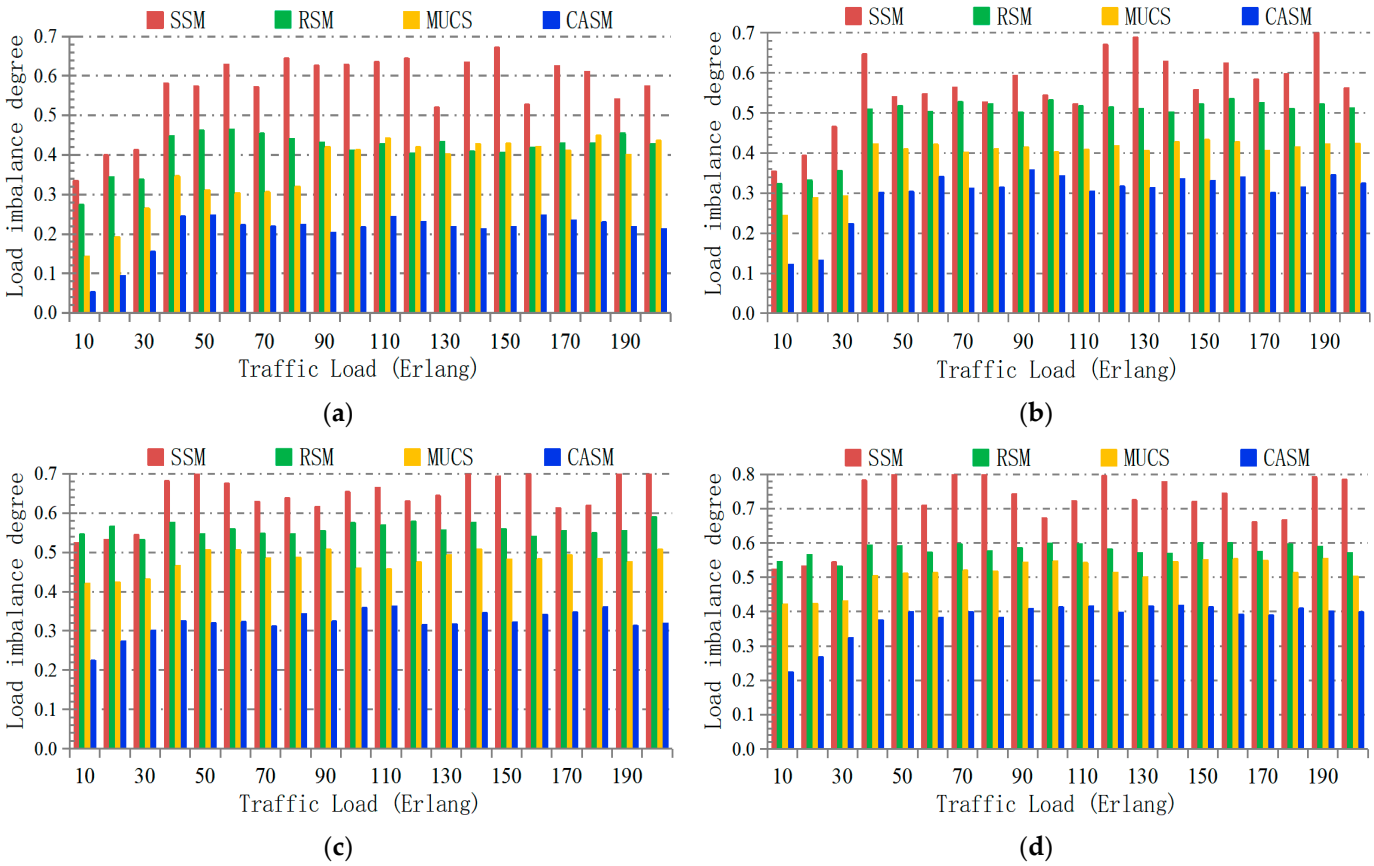

Figure 4. Load imbalance degree of different schemes under different topologies: (**a**) NSFNET Topology; (**b**) OS3E Topology; (**c**) Ta2 Topology; (**d**) Interoute Topology.

For the migration cost of the switch, under different network topologies, we analyze and compare the migration cost indicators of different schemes, as shown in Figure 5. Under different traffic loads, the migration cost index of the proposed CASM scheme is much lower than that of the existing scheme. Since in the actual implementation process, the CASM scheme selects the best switch from the set of switches associated with the overloaded controller as the switch to be migrated by minimizing the migration cost when selecting switches, thereby reducing the additional migration cost generated by switch migration activities. In Figure 5a, we test the performance of the proposed CASM scheme based on the NSFNET topology. From the experimental data, it can be seen that compared with RSM and MUCS, the migration cost of the proposed CASM scheme is reduced by 45.9% and 55.3%, respectively. For other types of topologies, compared with RSM and MUCS schemes, the proposed CASM scheme reduces an average of 43.7% and 51.2%. Compared with the MUCS scheme, the CASM scheme reduces the migration cost by about 15%. Since during the switch selection process, the CASM scheme selects the best switch based on minimizing the migration cost and associates it with the target controller.

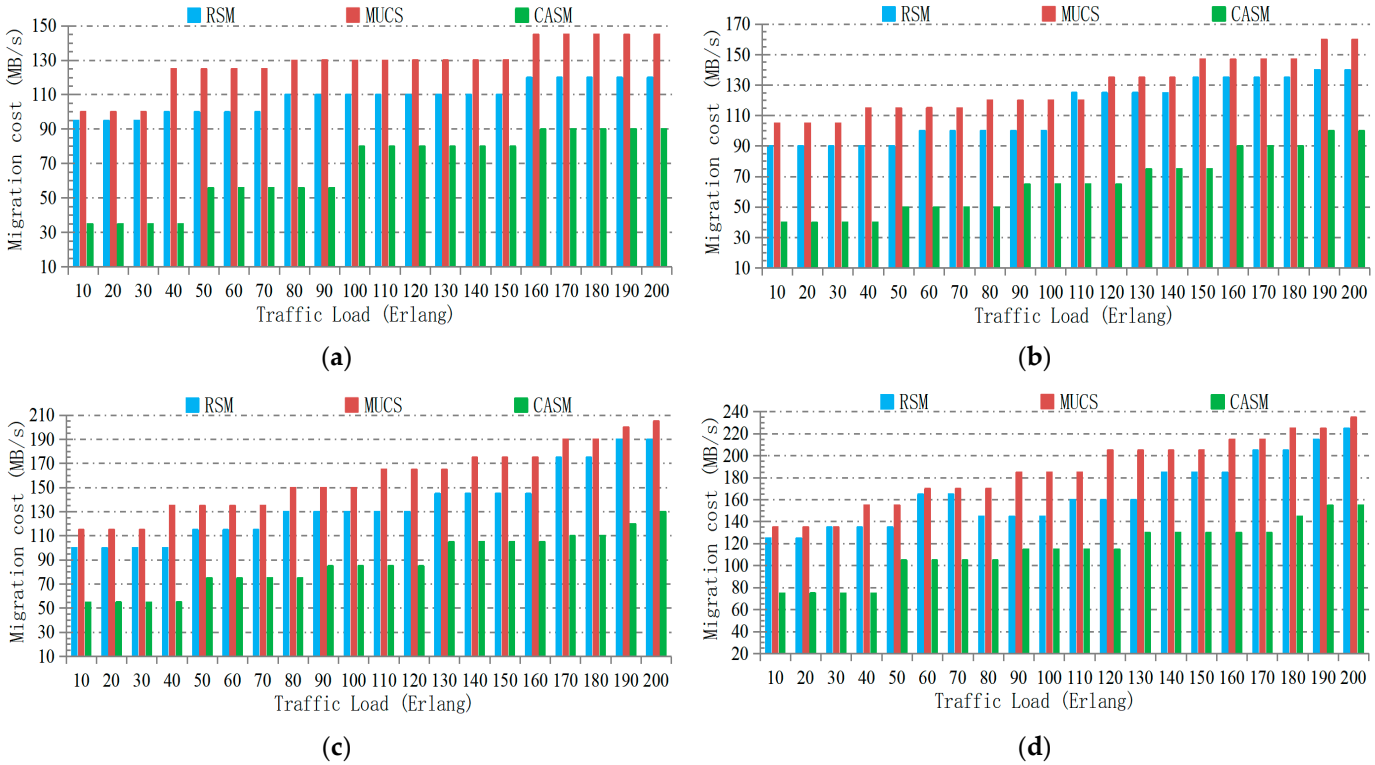

Figure 5. Migration cost of different schemes under different topologies: (**a**) NSFNET Topology; (**b**) OS3E Topology; (**c**) Ta2 Topology; (**d**) Interoute Topology.

5. Conclusions

In this paper, we propose a cost-aware switch migrate (CASM) strategy to achieve balanced distribution of controller load. Firstly, CASM calculates the load of the controller through multiple load indicators, and judges whether the controller is overloaded or underloaded based on the response time of the controller to the flow request message, so as to output the overloaded controller set and the underloaded controller set, respectively. Then the proposed scheme calculates the migration cost for each switch in the set of switches associated with the overload controller based on multiple migration cost indicators, and selects the optimal switch as the switch to be migrated based on the minimized migration cost. Finally, the CASM solution migrates the selected switches to be migrated from the domain network associated with the overloaded controller to the domain network associated with the underloaded controller, and the switch migration activity has been implemented. The performance evaluation shows that the proposed CASM is superior to the existing schemes in improving load balancing performance and reducing migration costs. In future research work, we will test and verify the performance of the proposed CASM scheme on a physical platform.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L. and Q.M.; validation, Y.L. and Z.S.; formal analysis, F.Y.; investigation, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L. and Q.M.; writing—review and editing, Y.L. and F.Y.; visualization, Z.S.; supervision, Y.L.; project administration, Y.L.; funding acquisition, Y.L. and F.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ22F010001, the Starting Research Funds from the Hangzhou Normal University under Grant 4115C50221204137 and 4085C50220204093, and the China Postdoctoral Science Foundation under Grant 2021M690179.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fiorani, M.; Tornatore, M.; Chen, L. Spatial Division Multiplexing for High Capacity Optical Interconnects in Modular Data Centers. *J. Opt. Commun. Netw.* **2017**, *9*, 143–153. [CrossRef]
2. Lin, R.; Chen, Y.; Wosinska, L. Disaggregated Data Centers: Challenges and Trade-offs. *IEEE Commun. Mag.* **2020**, *58*, 20–26. [CrossRef]
3. Chen, X.; Tornatore, M.; Zhu, S. Flexible Availability-Aware Differentiated Protection in Software-Defined Elastic Optical Networks. *J. Opt. Commun. Netw.* **2015**, *33*, 3872–3882. [CrossRef]
4. Yin, Y.; Zhang, H.; Zhang, M. Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks. *J. Opt. Commun. Netw.* **2013**, *5*, 100–106. [CrossRef]
5. Rival, O.; Morea, A. Elastic optical networks with 25–100 G format-versatile WDM transmission systems. In Proceedings of the OptoElectronics and Communications Conference, Sapporo, Japan, 5–9 July 2010; pp. 100–101.
6. Talebi, S.; Alam, F.; Katib, I. Spectrum management techniques for elastic optical networks: A survey. *Opt. Switch. Netw.* **2014**, *13*, 34–48. [CrossRef]
7. Chatterjee, B.; Sarma, N.; Oki, E. Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1776–1800. [CrossRef]
8. Velasco, L.; Vela, A.P.; Morales, F. Designing, operating, and reoptimizing elastic optical networks. *J. Lightwave Technol.* **2016**, *35*, 513–526. [CrossRef]
9. Gong, L.; Zhu, Z. Virtual optical network embedding (VONE) Over Elastic Optical Networks. *J. Lightwave Technol.* **2013**, *32*, 450–460. [CrossRef]
10. Zhao, B.; Chen, X.; Zhu, J. Survivable control plane establishment with live control service backup and migration in SD-EONs. *J. Opt. Commun. Netw.* **2016**, *8*, 371–381. [CrossRef]
11. Xu, Y.; Cell, M.; Wang, L. Dynamic Switch Migration in Distributed Software-Defined Networks to Achieve Controller Load Balance. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 515–529. [CrossRef]
12. Kilper, D.; Zhu, S.; Yu, J. Physical Layer Control for Disaggregated Optical Systems. In Proceedings of the Asia Communications and Photonics Conference (ACP), Hangzhou, China, 26–29 October 2018; pp. 1–3.
13. Zhu, Z.; Lu, W.; Zhang, L. Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing. *J. Lightwave Technol.* **2012**, *31*, 15–22. [CrossRef]
14. Gong, L.; Zhou, X.; Liu, X. Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks. *J. Opt. Commun. Netw.* **2013**, *5*, 836–847. [CrossRef]
15. Lu, P.; Zhang, L.; Liu, X. Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks. *IEEE Netw.* **2015**, *29*, 36–42. [CrossRef]
16. Dixit, A.; Hao, F.; Mukherjee, S. ElastiCon: An elastic distributed SDN controller. In Proceedings of the IEEE Symposium on Architectures for Networking and Communications Systems, Marina del Rey, CA, USA, 20–21 October 2014; pp. 17–27.
17. Heller, B.; Sherwood, R.; Mckeown, N. The controller placement problem. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012; pp. 7–12.
18. Chen, G.; Chen, H.; Wang, Z. DHA: Distributed decisions on the switch migration toward a scalable SDN control plane. In Proceedings of the IFIP Networking Conference, Toulouse, France, 20–22 May 2015; pp. 1–9.
19. Chen, H.; Chen, G.; Wang, Z. A game-theoretic approach to elastic control in software-defined networking. *China Commun.* **2016**, *13*, 103–109. [CrossRef]
20. Yu, J.; Wang, Y.; Pei, K. A load balancing mechanism for multiple SDN controllers based on load informing strategy. In Proceedings of the Asia Pacific Network Operation and Management Symposium, Kanazawa, Japan, 5–7 October 2016; pp. 1–4.
21. Cello, M.; Xu, Y.; Walid, A. BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration. In Proceedings of the IEEE International Conference on Cloud Engineering, Vancouver, BC, Canada, 4–7 April 2017; pp. 40–50.
22. Chen, G.; Chen, H.; Hu, H. Dynamic switch migration towards a scalable SDN control plane. *Int. J. Commun. Syst.* **2016**, *29*, 1482–1499. [CrossRef]
23. Cui, J.; Lu, Q.; Zhong, H. A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 1197–1206. [CrossRef]
24. Zhou, Y.; Wang, Y.; Yu, J. Load balancing for multiple controllers in SDN based on switches group. In Proceedings of the 19th Asia-Pacific Network Operations and Management, Seoul, Korea, 27–29 September 2017; pp. 227–230.
25. Sahoo, K.; Tiwary, M.; Sahoo, B. RTSM: Response time optimisation during switch migration in software-defined wide area network. *IET Wirel. Sens. Syst.* **2019**, *10*, 105–111. [CrossRef]
26. Wang, Y.; Tao, X.; He, Q. A dynamic load balancing method of cloud-center based on SDN. *China Commun.* **2016**, *13*, 130–137.
27. Hu, T.; Lan, J.; Zhang, J. EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 452–464. [CrossRef]

28. Hu, T.; Yi, P.; Zhang, J. A distributed decision mechanism for controller load balancing based on switch migration in SDN. *China Commun.* **2016**, *15*, 129–142. [CrossRef]
29. Liu, Y.; Gu, H.; Yu, X. Dynamic SDN Controller Placement in Elastic Optical Datacenter Networks. In Proceedings of the Asia Communications and Photonics Conference (ACP), Hangzhou, China, 26–29 October 2018; pp. 1–3.
30. Liu, Y.; Gu, H.; Wei, Q. Multi-Controller Placement Based on Load Balancing in Inter-DC Elastic Optical Networks. In Proceedings of the Asia Communications and Photonics Conference (ACP), Chengdu, China, 2–5 November 2019.
31. Liu, Y.; Gu, H.; Yan, F. Multi-Controller Placement Based on Two-Sided Matching in Inter-Datacenter Elastic Optical Networks. In Proceedings of the IEEE International Conference on Communications, Dublin, Ireland, 7–11 June 2020; pp. 1–6.
32. Liu, Y.; Gu, H.; Yan, F. Highly-Efficient Switch Migration for Controller Load Balancing in Elastic Optical Inter-Datacenter Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2748–2761. [CrossRef]
33. Tootoonchian, A.; Gorbunov, S.; Ganjali, Y. On controller performance in software-defined networks. In Proceedings of the Internet Cloud Enterprise Network Services, San Jose, CA, USA, 24 April 2012; pp. 10–15.
34. Yao, G.; Bi, J.; Li, Y. On the capacitated controller placement problem in software defined networks. *IEEE Commun. Lett.* **2014**, *18*, 1339–1342. [CrossRef]
35. Shakya, S.; Pradham, N.; Cao, X. Virtual network embedding and reconfiguration in elastic optical networks. In Proceedings of the IEEE Global Communication Conference, Austin, TX, USA, 8–12 December 2014; pp. 2160–2165.
36. Han, L.; Li, Z.; Liu, W. Minimum Control Latency of SDN Controller Placement. In Proceedings of the IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 2175–2180.
37. Knight, S.; Nguyen, H.; Falkner, N. The internet topology zoo. *J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]
38. Landi, G.; Capitani, M.; Kretsis, A. Inter-domain optimization and orchestration for optical datacenter networks. *J. Opt. Commun. Netw.* **2018**, *10*, 140–151. [CrossRef]