Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5207105, 10 pages
https://doi.org/10.1155/2020/5207105

Research Article

Improving Fairness for Distributed Interactive Applications in Software-Defined Networks

Ran Xu[1] **and Weiqiang Zhang**[2]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Industrial Robot Training Center, Tangshan Vocational and Technical College, Tangshan, China*

Correspondence should be addressed to Weiqiang Zhang; weiqiangzhang@yeah.net

Received 9 March 2020; Revised 4 July 2020; Accepted 28 July 2020; Published 12 August 2020

Guest Editor: Hou-Sheng Su

With the popularization of distributed interactive applications (DIAs), for getting good interactive experience among participants, efficient and fair allocation of network resource should be considered. In software-defined networks, the presence of central controllers provides novel solution to deploy customizable routing for interactive applications, which allows fine-grained resource allocation for DIAs to achieve fairness among participants. But opportunities always come with challenges, the wide spread user locations often require distribution of controllers to meet the requirements of applications. Hence, the latency involved among participants is directly affected by the processing time of controllers. In this context, we address the DIAs' fair resource provisioning problems on computing and links load with the objective of balancing the achievable request rate and fairness among multiple flows in SDN networks. We firstly formulate the problems as a combination of controller loading and routing optimization. Then, we propose proactive assignment controller algorithm based on deep learning and fairness path allocation algorithm to share the bottleneck links. Compared with the state-of-the-art greedy assignment algorithm and priority order allocating algorithm, the final result is proven to get better fairness on controller and link load among DIAs' participants by trace driven simulation.

1. Introduction

In recent years, smart devices have emerged, and a variety of distributed interactive applications (e.g., multiparty network gaming, online video conference, teleconferencing, online trading, and multiperson augmented/virtual reality) are provided on smart devices. More and more applications require cooperation. One example of real-world most popular games with over 300,000 daily players can be found in [1], which has been released as a mobile game in China. As time goes on, we will see the previously distributed research (e.g., distributed live music concerts [2], shared whiteboard, shared workspaces on collaborative design [3], web-based e-learning, and interactive desktop applications on distributed infrastructures [4]), which will be or has been already used on diverse devices [5], which makes good user experience become more and more important.

In the network, different styles of DIAs (e.g., game) have different thresholds for maximum tolerable delays [6, 7], and resource allocation subject to the varying bandwidth capacities of different links. The networks with a single bottleneck resource, where congestion is signaled by the build-up of a queue at the bottleneck's buffer and where propagation delays are significant [8, 9], show that the network does not converge to an efficient and fair equilibrium in networks with multiple bottleneck nodes. Hence, excessive unfairness for on-demand resource provisioning can severely degrade the participants quality of experience. The fairness problems of flows include the propagation delay as well as the sharing of bottleneck links to multipath communication. For example, when playing a game over a network where quality of service is not guaranteed, the positions of an object at different players' terminals may be disturbed owing to the network delay [10], and latency

differences between players can lead to unfairness in fast-paced First Person Shooter games or 3D virtual space on haptic interface devices [10, 11]. Moreover, upon reaching such an allocation, the same weight priority for the requests 10 Gb/s and 2 Gb/s of bandwidth may drop sharply to the first service. To solve these problems, cloud computing and other techniques enable users to process their applications on the server-side [12]; for instance, the effort in [13] provided mechanisms and methods for facilitating tiered service model-based fair allocation of resources for application servers in multitenant environments. However, server-side latency compensation causes CPU processing power and memory overhead, which will restrain the total number of customers on the application servers. In addition, the efforts of previous works in [10, 14, 15] have proven that one of the contributing factors to the inconsistencies experienced is network, server-side estimation, which is usually deduced from application behaviors that are not always concerned to network issues, and the network in communication remains as a major barrier to achieve high quality interaction experience even if there is no limitation on the availability of server capacities [5, 10, 12, 16].

In recent years, a new paradigm that is known as Software Defined Networks (SDN) emerged [17]. SDN gives possibilities principled solution to manage the networks behavior at runtime for interactive online applications [18]. The basic concept of SDN is controller to run on servers and the network devices forward traffic based on installed flexible rules. Interactive applications can utilize this state information to obtain estimates of available bandwidth and latency that are current and more accurate than what can be obtained, since they are measured in real-time by the SDN controller [19]. Therefore, rerouting and reconfiguration are potentially fast and efficient in SDN [20]. This enables traffic to be steered in real-time. Whereas many interactive applications (e.g., smart homes, IoT applications, and edge computing) are widely distributed in different geographical areas, for better utilization of controller resources, it requires distribution of controllers process effectively. And the controller long processing time may not be significant for elephant flows; however, when the (1) new flow arrives, (2) logical topology changes, and (3) link failures and the intermediate switches can not resolve the data package; they will intercept the data flow and forward Packet-In message to the SDN controller [21]. The control requests are one of the major contributors of messages received by the controller [22]. Even if the forwarding path has been established, usually the network requests are frequent and rule storage space is limited; it still has the possibility of sending requests to the controller [21]. In terms of geographical distribution of the sources of demand, long processing time will fundamentally limit the network's ability to quickly react to events such as link congestion or failures [23, 24]. Hence, as part of the network, when we address the distributed interactive network applications (DIAs) fair allocation resources between various flows, the controllers' high processing time must be prevented.

Based on the blueprint of the software-defined network innovation, in this work, we investigate the flows fair allocation problem combined with the problem of control processing and routing in SDN. Due to the latency involved in the interaction, in the distributed SDN architecture, in order to achieve fairness allocation for user flows, a runtime scheduler is needed to answer the following questions: (1) How to dynamically map requests to the controllers? (2) Which routes shall be used to interconnect the communication, and how to allocate the multiple paths and share the limit bandwidth capacity fairly?

Specifically, the summaries of our contributions are listed as follows:

(1) We present a fair resource allocation for distributed interactive applications in Software-Defined Networks, in which the structural controller is introduced. It can be viewed as the flows fair sharing processing and routing resources.

(2) We solve controller assignment problem proactively based on deep learning and the adjustment based on water filling process. From the view of routing and bandwidth allocation, we consider the multipath and bottleneck to fairly distribute links.

(3) We choose some representative data sets to conduct some experiments. The experimental results show that the schemes get better fairness, while achieving better performance in terms of reaction time, etc.

The rest of this paper is organized as follows. In Section 2, we describe the problems in detail and model the problem. In Section 3, proactive controller assignment algorithm and routing and bandwidth's fair path allocation algorithm with the example are given. In Section 4, we choose some commonly used data sets to test the presented algorithms' performance. Moreover, we also compare performance of algorithms with others. In the final section, we give some conclusions.

2. Problems Description and Modeling

2.1. Processing and Routing. The typical unfairness of DIAs among participants is illustrated in Figure 1. Assuming there is an application for user 1 and user 2, both of them need to send real-time packets to the interactive application server, and the objective is to fairly allocate path delay between them. Most of the traditional routing protocols prefer to adopt the low latency paths to deliver applications traffic to destinations. Generally, the classic shortest path routing protocols (e.g., OSPF) are always used as the autonomous system routing protocol in previous network, even if, regardless of links bandwidth capacity limitations and controllers processing, in Figure 1, we suppose that users need to transfer one unit size of information and use the link propagation delays as the OSPF weights: (a) if the default paths for user 1 and user 2 are routing based on OSPF weights, there are user 1 $\longrightarrow$ S1 $\longrightarrow$ S2 $\longrightarrow$ S8 $\longrightarrow$ S10 and user 2 $\longrightarrow$ S4 $\longrightarrow$ S7 $\longrightarrow$ S9 $\longrightarrow$ S10, and the latency difference is 4; (b) if it is based on the paths that are denoted by the arrow lines in the picture, the latency difference will be 0; thus, the equalizing application delays between user 1 and

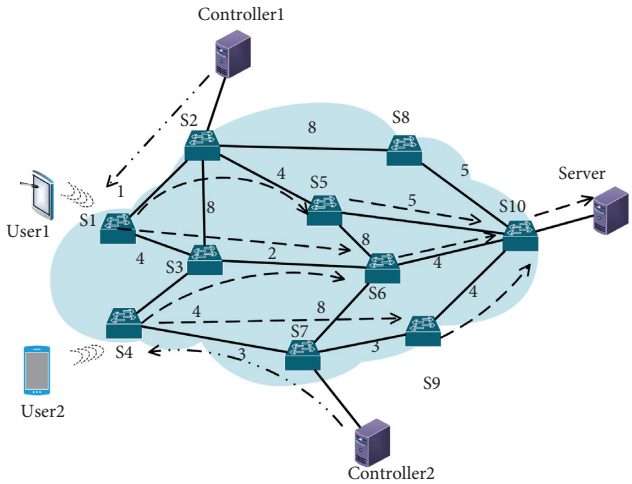

Figure 1: Illustration of allocation in SDN network.

user 2 are unfair in the shortest path routing way. In addition, different styles of DIAs have independently accounted for bandwidth, latency, and loss, and interaction process occupies the scarce bandwidth resources; how to adjust weights or priorities to flexibly allocate shared bottleneck links? For example, we may want to give a high priority user a weight of 10.0, which has been allocated 1 Gb/s, and a weight of 1.0 to another, which is allocated 2 Gb/s, when they share the bottleneck link.

Moreover, for distributed interactive applications in SDN, the customers carry on mutual interactions among themselves. The interaction includes not only the network latencies from the customers to server but the latencies between the customers and the controllers. Figures 2(a) and 2(b) show the structure and the interactive process in SDN network. In the figures, we refer to the request process as control and interaction. When issuing an operation, the DIA clients send action to the switch, which detects the DIA event, its relays are assigned a controller for further processing, the controller will compute and synchronize the state, and the participants will communicate through the fabric network. Thus, the controller's performance is based on the control traffic overhead, which will influence all users. In principle, SDN is possibly principled to provide customized service; therefore, it is feasible to associate a controller such that the processing time influencing fairness and consistency for all users is minimum.

2.2. Problem Formulation. The first thing we model the network as an undirected graph $G = (V, E)$, where E states the set of links with link bandwidth b_e per link e, and V represents the set of nodes containing the switches, servers, and the controllers. We use N to denote the number of controllers, and c_j represents the processing capacity of controller j. There is an interval of time slot $t \in \{1, 2, \ldots, T\}$ with a set F of flows and d denotes the delay between source user to its associated destination server, and corresponding flow $f_i \in F$ starts from source node and finally reaches destination node. The main symbols used throughout this paper are listed in Table 1.

We consider a discrete slot system model where the switch requests can be recorded and the controller provisioning decisions can be updated. At time slot t, function $a_i(t)$ denotes the traffic rate at switch i, and the requests are aggregated at the processing queue of the connected controllers. The controller can be modeled as an M/M/1 queue. Here, we use the weighted average of controller response time, which evaluates the performance of controller relative to the proportional of its load, which has been proven in [25] and described below. The load of controller j is denoted as

$$l_j(t) = \sum_{i=1}^{|V|} w_i a_i(t) x_{ij}(t), \tag{1}$$

where $x_{ij}(t)$ is denoted as whether switch i is connected to controller j. Symbol w_i represents the weight of traffic to requests. By applying the Little's law, the average processing time of the controller j can be represented as

$$p_j(t) = \frac{1}{c_j - l_j(t)} O\left(|V|^2\right). \tag{2}$$

So the overall controller response time can be calculated as

$$R(t) = \frac{\sum_{j=1}^{N} l_j(t) p_j(t)}{\sum_{j=1}^{N} l_j(t)}. \tag{3}$$

Since multiple DIAs user groups request the bandwidth and they share the same physical network, the bandwidth consumption may not exhaust the available link bandwidth. For the realistic network conditions, and the distributed interactive applications (DIAs) such as the bandwidth-intensive tasks (e.g., streaming game screens to clients) in [5], to avoid unfairness to use the resources, we need to ensure that service is distributed among users in a fair manner. In the most basic form, each network flow is associated with a utility as a function of its rate, various notions of fairness can be expressed simply by changing the shape of the utility functions [26]. According to [9], the utility function of interaction flow is defined as

$$u_{f_i}\left(r_{f_i}\right) = \frac{r_{f_i}^{1-\alpha}}{1-\alpha}, \tag{4}$$

so the overall utility can be calculated as

$$U = \sum_{f_i \in F} u_{f_i}\left(r_{f_i}\right), \tag{5}$$

An operator enables to express different preferences on the fairness/efficiency trade-off curve by varying α; when $\alpha = 2$, it translates to the minimum potential delay fairness, which can be interpreted as the delay of user transferring a file of unit size with the rate allocated of r_{f_i} going through allocated path p_{f_i}, eventually converging to the egalitarian max-min fair allocation as $\alpha \longrightarrow \infty$, and $\alpha \approx 5$ is sufficient for very good approximation in [26]. Thus, the object of optimization can be expressed as

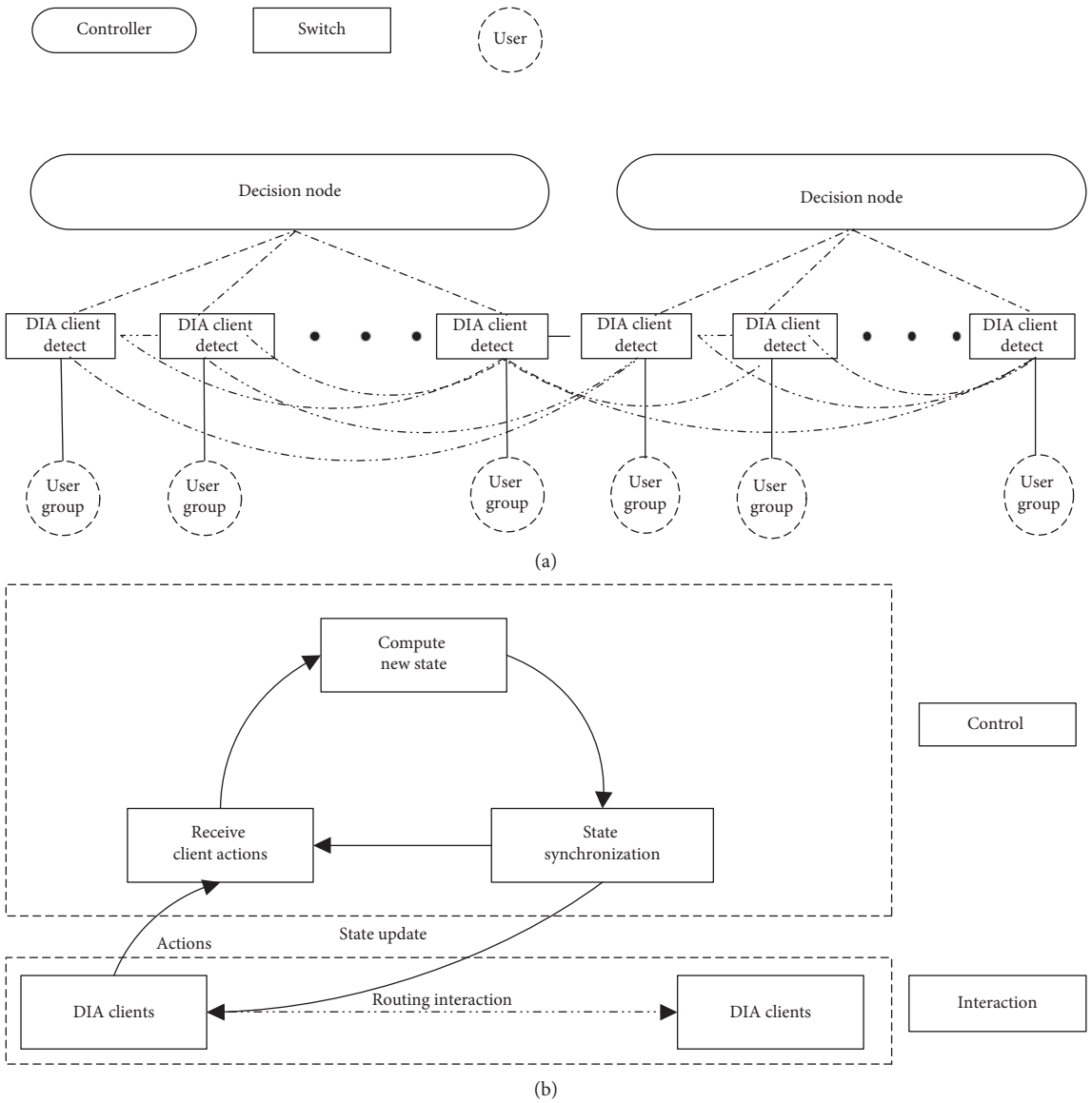

Figure 2: The structure and the interactive process in SDN network.

Table 1: Key symbol.

Symbol	Definitions or descriptions
$G = (V, E)$	The graph representing the network
N	The number of controllers
c_j	The processing capacity of controller j
t	The time slot
$a_i(t)$	The traffic rate at switch i
$l_j(t)$	The load of controller j in time slot t
$p_j(t)$	The processing time of the controller j in time slot t
α	A nonnegative variable constant
$x_{ij}(t)$	A binary variable indicating whether switch i is connected to controller j
x_{ihj}	A binary variable indicating the switch h is included in the path p_{ij} between source node i and destination j

$$
\begin{aligned}
\min \delta \quad &= \sum_{t=1}^{T} (R(t) - U(t)) \\
&\sum_{f_i \in F, e \in p_{f_i}} r_{f_i} \le b_e, \quad \forall e \in E \quad (1) \\
\text{s.t.} \quad &(2) l_j \le c_j, \quad \forall j, t \quad (2) \\
&d_{f_i}(t) < d_i^m, \quad \forall f_i \in F, t \quad (3) \\
&x_{ihj} \le 1, \quad \forall p_{ij} \in p_f, f \in F \quad (4),
\end{aligned} \tag{6}
$$

where equation (1) reflects the amount of routing allocation flow that can not exceed the link capacity. Constraint (2) specifies that no controller is overloaded and constraint (3) denotes that any assigned path delay should be less than the maximum tolerable delays d_i^m. We define x_{ihj} to represent

that the switch h is included in the path p_{ij} between source node i and destination j, and constraint (4) guarantees loop-freedom when a switch is only used once per path.

3. Allocation Algorithm

3.1. Proactive Controller Assignment Algorithm. In this section, we firstly use controller and switch to examine and identify controller impacts. We randomly generated 60 groups of users; in every group, there are five pairs of users that communicate with each other for 2 seconds. For test 1 and test 2, the demands of bandwidth are 1 Mb and 2 Mb, respectively. When users contact with each other in SDN, Table 2 displays the number of packages, which shows that communication with controllers is inevitable, and the controller will bring extra processing latency to the interaction, and the request rate is monotonically increasing with the number of requests. Although the cost function of the response time can take any form to model a provider's specific delay cost factors [25], here, we only require to be monotonically increasing with the number of requests.

For faster controller response and better utilization of controller resources, we can dynamically allocate the control domain, and then, when requests vary, the switches are dynamically mapped to controllers. Hence, to figure out the controller assignment problem, the method such as DCAP in [25] can be used for data center, or scheme in [16, 27] used for large scale network, but we seek a proactive approach ahead of time. Therefore, we use the deep learning to estimate requests for user group based on their historical usage; the details of deep learning can be seen in our previous research [28]. Moreover, inspired by [27], but different from adjusting the facility location, we estimate the aggregate demand and adjust the control domain with the fixed number and location of controllers.

As shown in Algorithm 1, first, the method SM in [25] is used to generate a stable matching between switches and controllers. Second, since all requests that already have a promised value from the previous time-step, to reduce the computational overhead and improve response time, we examine the solution from the current time-step and future time-steps. Our algorithm conducts the increasing loading heuristic in which the highest loading values are decreased to the lowest loading values similar to a water filling process. The demands used in algorithm are based on peak historical usage, and different flow groups can peak at different times. Besides, these schemes plan for the time-step can be set to several minutes similar to Google [29] and Mircosoft [30].

3.2. Routing and Bandwidth Allocation Algorithm. One of the challenges is the sources determining the paths of traffic flows based on the aggregate propagation delay along path, and the optimization for bandwidth allocation to individual flows. Here, we present a fairness objective scheme for user groups, which share the bottleneck links for multipath communication. As shown in Algorithm 2, traffic is spread across multiple paths and ensures that users fairly share the requests served that enable the network to run at higher levels of utilization.

Table 2: Formal context.

Name	Total	Controller
Test 1	104076	21828
Test 2	154389	21621

Since different styles of DIAs have different thresholds for maximum tolerable delays, these thresholds influence the user rating, which can be further adjusted according to the user demand sensitivity or operating experience [31], even if we consider user-defined fair allocation of spare bandwidth, while still ensuring minimum bandwidth guarantees for each client [32]. In the algorithm, first, we use an adaptation of the gradient descent algorithm for flows to determine the paths, and the appropriate fraction on each path, so as to achieve weighted fairness for different service-level demands. In other words, we now try to maximize $\sum_{f_i \in F} u_{f_i}(r_{f_i}) - \sum_{i \in P_i D}(d_i^s / |L(i)|)$. Here, $|L(i)|$ denotes the number of links in flow i' s path, and $P_i D$ is the set of flow paths in the same group. The latter part of the equation is a "penalty" function, which penalizes the latency difference path, since the d_i^s and $|L(i)|$ are integers, and the $d_i^s \geq 1$ in each link. We estimate flow group demand by usage history x_i, which can predict the rate by the deep learning. Hence, the optimal value of weight w_i is given by $w_i = x_i U_i'(x_i)$. Since the weight is proportional to the bid, the ratio of w_i to x_i is similar to the price per unit. This has an intuitive interpretation: it sets the fraction of each path, such that the latency difference cost is equal to the overall sum of delay of the links on flow path, the weight of which is equal to the marginal utility as far as the price of per unit resource. If the values are at the optimal values, this calculation gives the optimal rates, but generally, the calculated values are not optimal rates for incorrect demand. After that, we execute a second optimization that uses an adaptation of MPFA [33] algorithm to find the bottleneck and seek the global max-min fairness by allocating all of the unused capacity in the current time-step. Here, using the algorithm has a crucial advantage: it finds a feasible and efficient allocation approximately, even if the estimated rates are not optimal.

An example is used to help illustrate the algorithm comprehension. Figure 3 is a simple network with three links, e_1, e_2, and e_3, with bandwidth numbers capacity 10, 11, and 2, respectively. Assume that there are three flows: the source-destination pairs of f_1 and f_2 are from node 1 to 3, and f_3 is from node 1 to 2. For simplicity of calculation, each link has a delay value of 1, and all flows have demand of 13. First of all, f_2 is more sensitive on path delay than f_1 (e.g., the first person shooter and role playing game); in other words, pay more than f_2; the flows f_1, f_2 and f_3 have weights of 1, 2, and 3, respectively, denoting the flow of relative priority. Hence, calculated by algorithm, the paths for f_1 are e_2 and $e_1 \longrightarrow e_3$ (the fractions of multipath are 0.67 and 0.33, respectively), f_2 is e_2, and f_3 is e_1. After determining the paths and fraction of flows, the algorithm

(1) Require: c_j: processing capacity of each controller;
(2) Ensure: mapping between switches and controllers $x_{ij}(t)$
(3) Initialize: initial Stable Matching
(4) for $t = 1, 2, \ldots, T$ do
(5) estimated request arrival rate $a_i(t+1)$;
(6) sort the nodes in V to decreasing order C_s according
(7) to traffic overhead of current time-step t and next
(8) time-step $t+1$;
(9) Temporarily holds the proposal, each switch builds list
(10) w_s according to the worst response time;
(11) for $i = 1, 2, \ldots, [|V|/2]$ do;
(12) Find the lowest loading node v of uıs neighbors
(13) here u is the first node in the ordered list C_s;
(14) mapping the smallest switch $j \in w_s \subset v$ to v;
(15) end for
(16) end for

Algorithm 1: Proactive assignment algorithm.

(1) Input: c_j: Request flow $f_i \in F$; allocable bandwidth
(2) capacities and delay for link $\forall e_k, \forall d_k$;
(3) Output: Allocated fair share for $\forall f_i$
(4) for each f_i in the user group do
(5) Find the paths and available multipath $P_i D$
(6) with the links $\min \sum_{i \in P_i D} (d_i^s / |L(i)|)$ on the flow paths $p_i \in P_i$
(7) within the maximum tolerable delays d_i^m;
(8) Calculate $w_i = x_i U_i'(x_i)$, the sum of
(9) multipath's weight $sw = \sum_{p_i \in P_i D} \min_{i \in S(i)} w_i$, $S(i)$ is the set
(10) of flows incident on link i, fraction $(i) = (w_i / sw)$
(11) end for
(12) while $\exists f_i$ do
(13) for each link e_i do
(14) Calculate the allocation function by $A_{e_k} = b_k$
(15) Calculate the fair share results of $(s_1, \ldots, s_i)$
(16) Sort the result and find the bottleneck
(17) e_j, $j \in \min(s_1, \ldots, s_i)$
(18) Upgrade the equation $A_{e_k} = b_k$ by $\min(s_1, \ldots, s_i)$
(19) end for
(20) Fair share allocation $af(f_i) = s_i \text{weight}_i$
(21) end while

Algorithm 2: Fairness path allocation algorithm.

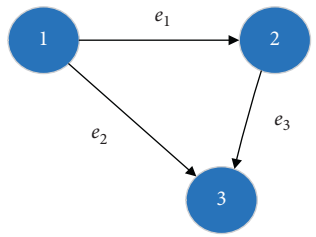

Figure 3: A simple example with three links.

executes a global allocation to fair share the bandwidth capacity. Based on the paths and fraction, the process is as follows:

Fist, find the bottleneck for each link, and the weighted bandwidth allocation function for links as

$$\begin{aligned} A_{e_1} &= 0.33(\min(13, s)) + \min(13, 3s), \\ A_{e_2} &= 0.67(\min(13, s)) + \min(13, 2s), \\ A_{e_3} &= 0.33(\min(13, s)), \end{aligned} \tag{7}$$

Calculated by $A_{e_k} = b_k$, the fair share results of s_{e_1} s_{e_2} and s_{e_3} roughly equal 3, 4.12, and 6, respectively. So, the e_1 is the bottleneck and f_1, f_3 fairly share the bottleneck of 3. And then, for equation $A_{e_k} = b_k$, the interval of s is based on the new values. The results of s_{e_2} roughly equal 4.5 and s_{e_3} is huge. Hence, l_2 is the bottleneck and f_2 shares fairly 4.5, and

the corresponding bandwidth allocation of (f_1, f_2, f_3) is $(3, 9, 9)$.

We now give a brief complexity analysis for algorithms presented above. For Algorithm 1, sorting the switches needs $O(N\log2(V))$ computation. One-slot computation needs to calculate N different numbers of active controllers. Thus, in $|N|$ mapping iteration, one controller accepts its most preferred switch. Therefore, the time complexity of Algorithm 1 is $O(|N|V\log2(V))$. Since the length of the whole time frame T and the complexity of calculating is $O(|T\|N|V\log2(V))$, the $|N|$ and $|T|$ are settable constants. For Algorithm 2, the complexity of calculating multipath path is $O(|E|)^2$, which records after the calculation and does not participate in the algorithm run any further. From the controllers' perspective, the complexity comes from searching for bottleneck fair share, and in each round, the search space is $O(\log2(E))$, and the complexity of calculating is $O(E\log2(E))$. Thus, the hierarchical two-phase algorithm's computation complexity is $O(|E|)^2 + O(|T|E\log2(E))$.

4. Experimental Evaluation

4.1. Simulation Settings. In our simulations, we demonstrate the rationale and advantages of the proposed mechanism by case study. The service demands are derived from the Skype traces [34], which contain timestamp, source, and destination pair and other information from real users, and we use the large topology with 115 nodes and 153 links obtained from [35]. We have extracted 290, 148 network flows from the dataset and used TensorFlow to implement the models, and the python package scikit-learn to calculate performance metrics. The detail of deep learning can be seen in our previous research [28]. Same as the setting of [36], here, we set the capacity of each controller as 1800 k flows/s. We use OpenDaylight and Mininet to test and identify controller impacts. All the simulations are carried out with six 2.2 GHz CPU cores and NVIDIA GeForce GTX 1060 GPU and 24 GB memories and Matlab 2015b. In each time slot, we regard the packets that have the same source and destination IP as a flow. The flow rate is calculated by dividing the total traffic size of the records in the flow by the interval of a time slot. We use the same α for each flow. The statistical result of service demands is about 10,000 flows per minute. Each link exhibits the same bandwidth resource capacity 150. We set a mean arrival rate 100. We set each flow's max packet arrival rate, and the number of arrival packets at each time slot is uniformly and randomly distributed with [0, 200]. First, we test the availability of the proactive approach, and then we use the greedy state-of-the-art algorithm from [36], which is designed for controller to be compared with the proactive assignment algorithm (PAA). Second, we evaluate our fairness path allocation algorithm (FPAA) through numerical simulations, and each flow's delay tolerance is set as a linear function with the type of different DIA task.

4.2. Effectiveness of PAA. In this section, we conduct simulations of different algorithms to test the strategy of request assign to controller. Although it is impractical that requests are triggered by each flow arrival event, the study in [25] has demonstrated the monotonically increasing relationship between the request number and the response time. Hence, in order to focus on evaluating the different algorithms strategy, here, we run with 5 controllers and evaluate the request arrival rates as load conditions.

To detect the availability of the proactive approach, we first examine the effectiveness of the request prediction. In Figure 4, to predict the 200 requests, we replace the source-destination requests pair with numbers, and the results show that deep learning achieves good accuracy on predicting the real request demands. Even though inaccurate prediction is inevitable, IP is usually segmented in different areas; therefore, the prediction error within a certain range is acceptable.

Then, we study the impact of different algorithms strategy for controller assignment. In each experiment, we simulate groups of DIA clients to communicate with each other. We compare the PAA algorithm against dynamic controller provisioning with greedy knapsack (DCP-gk) and nearest controller selection (NS) algorithm. Figure 5 shows single time slot to evaluate the load balancing and response time reduction among controllers. In this setting, we run all algorithms with sum of 5500 to 6500 k request rates to follow real-world data, which can be viewed as individual runs with different input requests in discrete time slots. The well-known Jain's index [37] is used to quantify the fairness between flows. Figure 6 shows the result of fairness for customers in the number of 20 time slots. We can observe that NS is sensitive to traffic variations due to lack of adjustment strategy, and DCP-gk uses the greedy algorithm to achieve a more balanced distribution. But compared with them, PAA predicts the request rates of flows that adopts an adaptive loading strategy by embracing the learning algorithms ahead of time, which achieves a good response time and fairness.

4.3. Effectiveness of FPAA. Next, in this section, we evaluate our path allocation algorithm for fair allocation and high utilization. We set each link the same bandwidth resource capacity, and each flow's delay tolerance related to the type of different DIA task (for example, game video encoding is less sensitive than streaming the game screens to clients). Our simulation spans over a few of 5-minute time slots. At each time slot, the demands of flows create a severe bottleneck that shows bandwidth resource competition among the flows. Obviously, different types of DIAs have varied bandwidth demands. Here, we compare with prior work that usually corresponds to greedy allocation, such as [38] that ensures allocating demands in priority order. The priority order setting is based on the sensitive of delay. In addition, for comparison, we use a single path (SP) method that meets the demands as much as possible. Figure 7 denotes the result of fairness, and the allocation is extremely fair across all flows (source-destination pairs). The flow-level fairness is a direct consequence of sharing a bottleneck link to multipath cluster-to-cluster communication, and the performance of FPAA

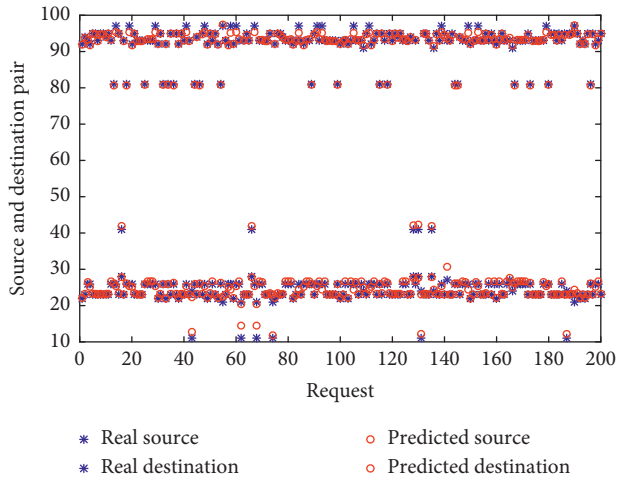

Figure 4: Performance of the proactive request.

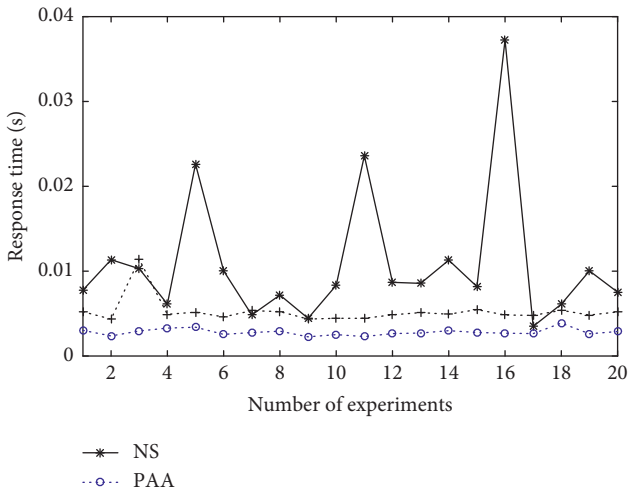

Figure 5: Response time.

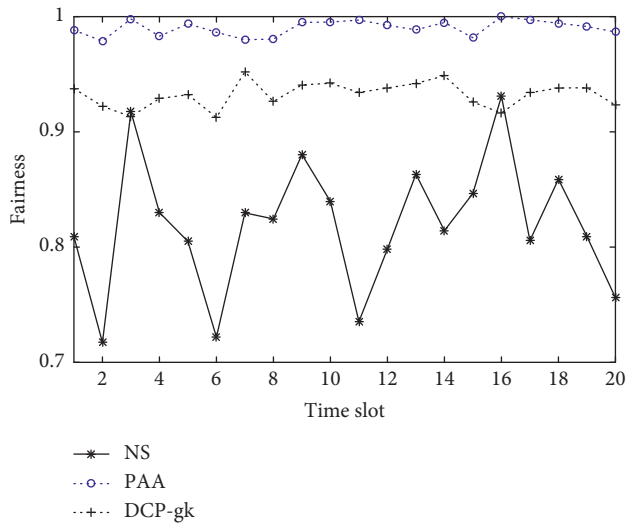

Figure 6: Fairness.

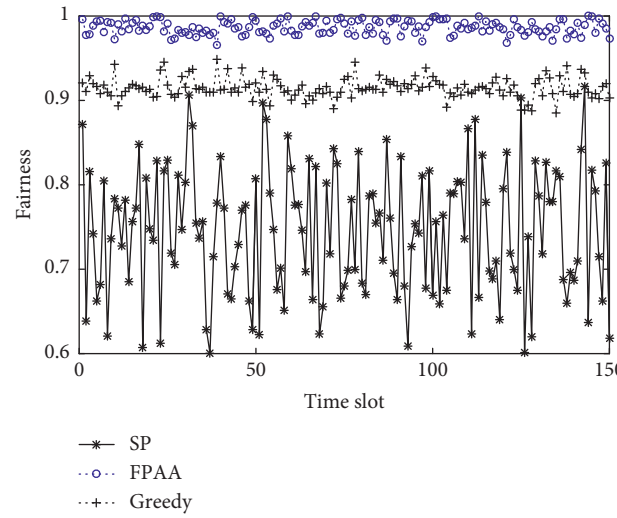

Figure 7: Fairness.

on routing allocation and fairness is significantly better on fairness of fair share.

5. Related Work

For distributed control plane across a cluster of controllers, the static mapping between switches and controllers improves robustness and scalability. Nevertheless, that may cause hot spots, which motivates several dynamic controller assignment works [25, 36, 39]. The work [39] firstly proposes a live switch migration protocol while introducing little overhead to the network. And work [36] presents an algorithm for dynamically and efficiently provision controllers in a WAN by periodically reassigning switches to controllers. However, they do not consider the processing time on controllers, which is a major cost in flow setup time [39]. The proposed heuristic is time-consuming and not partitioning application states and exploring the dependency between switches and applications. Work in [25] introduces the maintenance cost of the controller cluster and formulates the controller assignment problem as a long-term online cost minimization, which does not consider allocating bandwidth among users. Recent effort such as SWAN [38] is closely related but focuses on the network and routing infrastructure to effectively scale and utilize employing Software Defined Networking [33]. Previous studies have leveraged SDN capabilities for optimal bandwidth allocation in applications such as stream analytics [40]. However, the fairness considerations of SDN controller assignment and allocating bandwidth among users are not included.

A single controller is a single point of failure, and hence the distributed controller platform of the control plane has a fault-tolerant SDN controller that includes synchronization mechanisms, which is a well addressed. In several previous works, one of the first and most relevant proposals is ONIX [41], which provides a general framework where several important open issues have been identified, such as the trade-off between consistency, durability, and scalability. Other widely used distributed fault-tolerant controller

proposals are Hyperflow and ONOS [42, 43]. On the other hand, in the literature of fault-tolerant SDN controllers, Master-Slave controllers are a subclass that address the problem, where the central unit (the master) is in charge of taking the decisions, for example, [44, 45], where active and passive replication methods are proposed to provide fault tolerance. In the future, we need to make improvements based on previous achievements, but this is not in this work's scope.

In this work, we include fairness considerations as the state of the art of SDN controller assignment and allocating bandwidth among users in emerging DIAs environments. We generalize these concepts to the case of the problem. Building on related efforts of utility functions, weighted fair sharing, and controller-based control, we extend the problem for multipath routing and hierarchical fairness. This provides an elastic framework for resource sharing, with a significant degree of flexibility in balancing network bandwidth and controller processing power resources.

6. Conclusion and Future Work

The SDN architecture provides a pathway for networks to support scalable and adjustable multiparty interactive applications. In this paper, we have investigated the control and path unfairness problem to enhance the quality of the interactivity performance of DIAs. We have formulated the problem as a combinational optimization problem under SDN setting. The heuristic algorithms have been proposed to solve the problems, and real data is used to experimentally evaluate their performance. The results have shown that our scheme can achieve good performance. We further speculate that scheme can support different optimization policies that include reacting to failures and redirecting the control traffic.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] V. R. Carvalho, M. Lease, and E. Yilmaz, "Crowdsourcing for search evaluation," *ACM SIGIR Forum*, vol. 44, no. 2, pp. 17–22, 2011.

[2] A. Kapur, G. Wang, P. Davidson, and P. R. Cook, "Interactive network performance: a dream worth dreaming?" *Organised Sound*, vol. 10, no. 3, pp. 209–219, 2005.

[3] F. Liu, S. Xia, H. Shen, and C. Sun, "CoMaya: incorporating advanced collaboration capabilities into 3d digital media design tools," in *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work ACM*, pp. 5–8, San Diego, CA, USA, November 2008.

[4] T. Fu and G. Manglik, "Apparatus, systems and methods for deployment of interactive desktop applications on distributed infrastructures," US Patent 10003672, 2018.

[5] H. Wang, T. Li, R. Shea et al., "Toward cloud-based distributed interactive applications: measurement, modeling, and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 3–16, 2018.

[6] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance,?" *IEEE Network*, vol. 27, no. 4, pp. 16–21, 2013.

[7] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, 2014.

[8] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[9] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[10] M. Sithu, Y. Ishibashi, P. Huang, and N. Fukushima, "Influences of network delay on quality of experience for soft objects in networked real-time game with haptic sense," *International Journal of Communications, Network and System Sciences*, vol. 8, no. 11, pp. 440–445, 2015.

[11] S. Zander, I. Leeder, and G. Armitage, "Achieving fairness in multiplayer network games through automated latency balancing," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACM, Valencia, Spain, pp. 117–124, June 2005.

[12] H. Zheng and X. Tang, "Analysis of server provisioning for distributed interactive applications," *IEEE Transactions on Computers*, vol. 64, no. 10, pp. 2752–2766, 2015.

[13] X. Wang, "Facilitating tiered service model-based fair allocation of resources for application servers in multi-tenant environments," US Patent 10169090, 2019.

[14] M. Yu, M. Thottan, and L. Li, "Latency equalization as a new network service primitive," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 125–138, 2012.

[15] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of factors affecting players' performance and perception in multiplayer games," in *Proceedings of the 4th ACM SIGCOMM Workshop on Network and System Support for Games*, ACM, Hawthorne, NY, USA, pp. 1–7, October 2005.

[16] L. Zhang and X. Tang, "Optimizing client assignment for enhancing interactivity in distributed interactive applications," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1707–1720, 2012.

[17] C. Lin, J. Hu, G. Li, and L. Cui, "A review on the architecture of software defined network," *Chinese Journal of Electronics*, vol. 27, no. 6, pp. 1111–1117, 2018.

[18] S. Gorlatch and T. Humernbrum, "Enabling high-level QOS metrics for interactive online applications using SDN," in *Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 707–711, IEEE, Garden Grove, CA, USA, February 2015.

[19] A. T. Naman, Y. Wang, H. H. Gharakheili, V. Sivaraman, and D. Taubman, "Responsive high throughput congestion control for interactive applications over SDN-enabled networks," *Computer Networks*, vol. 134, pp. 152–166, 2018.

[20] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *Proceedings of the 2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–5, IEEE, Nassau, Bahamas, July 2013.

[21] M. Canini, D. Venzano, P. Peresini et al., "A nice way to test openflow applications," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, April 2012.

[22] K. S. Sahoo, M. Tiwary, B. Sahoo, R. Dash, and K. Naik, "DSSDN: demand-supply based load balancing in software defined wide area networks," *International Journal of Network Management*, vol. 28, no. 4, Article ID e2022, 2018.

[23] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: high-availability design principles drawn from googles network infrastructure," in *Proceedings of the ACM SIGCOMM*, pp. 58–72, New York, NY, USA, August 2016.

[24] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proceedings of the ACM SIGCOMM*, pp. 527–538, Chicago, IL, USA, August 2014.

[25] T. Wang, F. Liu, and H. Xu, "An efficient online algorithm for dynamic SDN controller assignment in data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, 2017.

[26] K. Nagaraj, D. Bharadia, H. Mao et al., "Numfabric: fast and flexible bandwidth allocation in datacenters," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ACM, Florianopolis, Brazil, pp. 188–201, August 2016.

[27] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed placement of service facilities in large-scale networks," in *Proceedings of the 2007—26th IEEE International Conference on Computer Communications*, pp. 2144–2152, IEEE, Barcelona, Spain, May 2007.

[28] R. Xu, "Proactive VNF scaling with heterogeneous cloud resources: fusing long short-term memory prediction and cooperative allocation," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4371056, 10 pages, 2020.

[29] C. Y. Hong, S. Mandal, M. Al-Fares et al., "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ACM, Budapest, Hungary, pp. 74–87, August 2018.

[30] S. Kandula, I. Menache, R. Schwartz et al., "Calendaring for wide area networks," in *Proceedings of the 2014 ACM conference on SIGCOMM—SIGCOMM'14*, vol. 44, no. 4, pp. 515–526, ACM, Chicago, IL, USA, August 2014.

[31] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hofeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *Proceedings of the Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 330–335, Seoul, South Korea, June 2011.

[32] M. Elahi, J. van Egmond, M. Wang, C. Williamson, and J.-F. Amiot, "Fair and efficient dynamic bandwidth allocation with OpenFlow," in *Proceedings of the Companion of the ACM/SPEC International Conference on Performance Engineering*, pp. 36-37, Edmonton, Canada, April 2020.

[33] A. Kumar, S. Jain, U. Naik et al., "BwE," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 1–14, 2015.

[34] TCP STatistic and Analysis Tool: Skype Traces, http://tstat.polito.it/traces-skype.shtml.

[35] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, ACM, New York, NY, USA, pp. 231–236, November 2002.

[36] M. F. Bari, X. Wang, and M. Huang, "Dynamic controller provisioning in software defined networks," in *Proceedings of the IEEE CNSM*, pp. 18–25, Zurich, Switzerland, October 2013.

[37] R. K. Jain, D. M. W. Chiu, and W. R. Hawe, *A Quantitative Measure of Fairness and Discrimination*, Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA, USA, 1984.

[38] C.-Y. Hong, S. Kandula, R. Mahajan et al., "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 15–26, 2013.

[39] A. Dixit, F. Hao, S. Mukherjee et al., "Towards an elastic distributed SDN controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp. 7–12, Hong Kong, China, August 2013.

[40] W. Aljoby, X. Wang, T. Fu, and R. Ma, "On SDN-enabled online and dynamic bandwidth allocation for stream analytics," in *Proceedings of the 2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pp. 209–219, Cambridge, UK, September 2018.

[41] T. Koponen, M. Casado, N. Gude et al., "ONIX: a distributed control platform for large-scale production networks," in *Proceedings of the OSDI*, vol. 10, Vancouver, Canada, October 2010.

[42] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, San Jose, CA, USA, April 2010.

[43] P. Berde, M. Gerola, J. Hart et al., "ONOS: towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ACM, Chicago, IL, USAACM, Chicago, IL, USA, August 2014.

[44] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "Resilience of SDNs based on active and passive replication mechanisms," in *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 2188–2193, IEEE, Atlanta, GA, USA, December 2013.

[45] N. Katta, H. Zhang, M. Freedman, and J. R. Ravana, "Controller fault-tolerance in software-defined networking," in *Proceedings of the SOSR*, Santa Clara, CA, USA, June 2015.