# Joint Traffic-Aware Consolidated Middleboxes Selection and Routing in Distributed SDNs

Guiyan Liu, Songtao Guo, *Senior Member, IEEE*, Baochun Li, *Fellow, IEEE*, and Chao Chen

***Abstract*—Software middlebox-based services can be flexibly managed by software defined networking (SDN) and network function virtualization (NFV). Meanwhile, traffic routing can be simplified and the number of routing rules in the SDN-enabled switches can be reduced through the consolidated middlebox model. However, different network functions in middleboxes may alter the volume of processed traffic, so high congestion may occur in specific bottleneck links if middlebox selection and traffic routing are not well jointly planned. Besides, in a statically switch-controller configured SDN, traffic dynamics will not only affect the link load in the data plane, but also pose a challenge to controller load balancing. Therefore, it's necessary to achieve better quality-of-service (QoS) performance in both control and data planes. In this article, we first formulate this problem as a joint traffic-aware consolidated middleboxes selection and routing (JTMSR) problem and prove its NP-hardness. Then, we design a two-phase algorithm to achieve the controller and link load balancing where the first phase is to redirect selected flows by applying wildcard rules and the second phase is to find fine-grained routing path by a rounding-based algorithm with bounded approximation factor. Finally, compared with the existing algorithms through extensive simulations, it demonstrates that our method has near-optimal controller load balancing and link load balancing performance and can improve response time by 9.7% compared with static scheme.**

***Index Terms*—Software defined networking, consolidated middlebox, load balancing, rounding, scalability.**

## I. Introduction

CURRENTLY, software defined networking (SDN) has been well accepted for network resource management owe to the separation of control and data planes, and distributed controller architectures have been widely adopted to improve the scalability of SDN [1]. To improve network performance and realize a series of network policies such as intrusion detection system, traffic engineering (TE), and quality-of-service (QoS) control, network service providers may apply middlebox based services such as firewall and deep packet inspection. However, traditional middlebox is a standalone physical device that typically performs one network function and difficult to add new functionality, so it is a significant challenge to manage middleboxes [2]. Along with network function virtualization (NFV), middleboxes are integrated and implemented into SDN-capable network [3]–[5], which provides the efficient and flexible management for software middleboxes by moving management functions out of the hardware and placing them in a centralized controller. Therefore, middlebox functions can be quickly and effectively established and added on demand in the network.

Currently, NFV framework can be divided into three forms including thread-based one, VM-based one and others such as element-based framework and host-based framework. CoMb [3] is a pioneer NFV framework that can host VNFs contained in threads while Slick [6] implemented VNFs as a chain of functions placed across the network. But VM-based NFV frameworks are studied most because they contain VNFs in VMs for isolation, which is an important property for both performance and cloud security [7]. Middleboxes can also be implemented using VMs that can be flexibly installed at the physical machines (PMs) and virtualization enables implementation of the consolidated middleboxes [3], where a flow receives all of its required service functions at a single machine. The consolidated middlebox model can simplify traffic routing and reduce the number of routing rules in the switches. Thus, numerous works such as optimization strategies for middlebox scheduling [4], [8]–[12] and placement [13]–[21] have been extensively done by research communities.

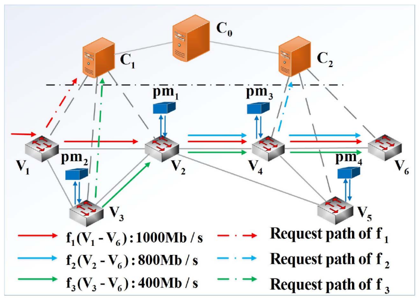
However, there still remain some challenges when scheduling middleboxes. Generally, the same type of middleboxes are usually deployed with multiple copies at various locations in a large-scale network, so the hardware resources such as CPU and memory cannot be amortized across multiple flows [22], [23]. Some existing works [9]–[12] designed and proposed middleboxes selection and routing strategies with different optimization objectives to avoid high congestion in specific bottleneck links, but when selecting middleboxes, one important point is neglected for these works that most middleboxes are traffic processing devices which may significantly change the volume of processed flows [13]–[15]. For example, Citrix CloudBridge WAN optimizer may compress traffic to 20% of its original volume before sending to the next hop and stateless transport tunneling (STT) proxy adds 76 bytes to each

Manuscript received March 20, 2020; revised October 5, 2020 and December 7, 2020; accepted December 22, 2020. Date of publication December 30, 2020; date of current version June 10, 2021. This work was supported by the National Key R&D Program of China (2018YFB0803400), the National Natural Science Foundation of China (No. 61772432, 61772433), the Technological Innovation and Application Demonstration Projects of Chongqing (cstc2018jszx-cyztzxX0014), the Fundamental Research Funds for the Central Universities (2019CDYGZD004) and Fundamental Research Funds for the Central Universities (2020CDCGJSJ071,2020CDCGJSJ038). The associate editor coordinating the review of this article and approving it for publication was C. Assi. *(Corresponding author: Songtao Guo.)*

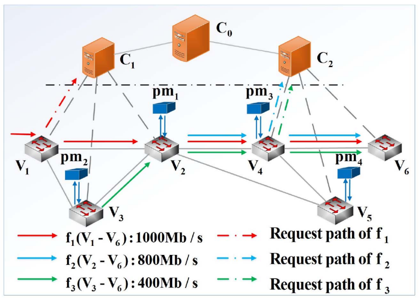
Guiyan Liu, Songtao Guo, and Chao Chen are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: songtao_guo@163.com).

Baochun Li is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S3G4, Canada.

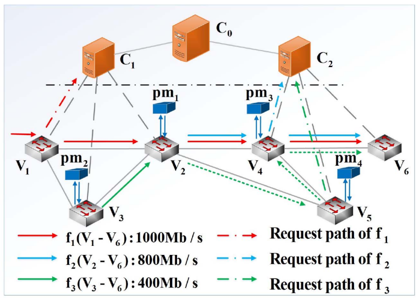
Digital Object Identifier 10.1109/TNSM.2020.3047391



(a) $f_3$ handled by $C_1$



(b) $f_3$ handled by $C_2$, $V_4$ reports to $C_2$



(c) $f_3$ handled by $C_2$, $V_5$ reports to $C_2$

Fig. 1. Illustration of different methods for routing $f_3$ to avoid a hot spot among controllers and link congestion in SDN middlebox-based network. The network is mainly composed of two parts: a set of controllers $\{C_0, C_1, C_2\}$ and six OpenFlow enabled switches $\{V_1, V_2, V_3, V_4, V_5, V_6\}$. PM is located at each switch via high-speed optical link. The processing capacity of controller is 1500 request time unit and there are three flows to be routed. The dotted grey line indicates the connection between the switch and its master controller. The blue, red and green lines respectively indicate the request paths of flow $f_1$, $f_2$, $f_3$. (a) $f_3$ handled by $C_1$ (b) $f_3$ handled by $C_2$, $V_4$ reports to $C_2$ (c) $f_3$ handled by $C_2$, $V_5$ reports to $C_2$.

processed packet. Another point is that these strategies mainly focused on optimizing QoS in a single controller which has limited processing capacity and easily becomes bottleneck of SDN, further blocking flows and degrading user experience.

To address above limitations, multiple controllers are used to cooperatively manage amount of switches/flows in the network to avoid single-controller failure and improve scalability. But one key challenge in multiple SDN controllers is how to avoid load imbalance caused by the uneven distribution of network traffic. Static mapping between switches and controllers results in long and highly varying controller response time [24], so dynamic assignment or switch migration [24]–[27] has been proposed to balance controller overhead, but the cost of switch migration is actually high (e.g., long latency and complex migration protocol) and frequent manipulation is not allowed [28]. To avoid switch migration cost, another way to overcome controller load imbalance is deploying wildcard rules on switches in advance to implement flow redirecting, which refers to redirect matched flows to other switches that will report the header of packets of this flow to the associated controllers [29]. In Fig. 1(a), $f_1$ and $f_3$ are handled by $C_1$ while $f_2$ can be handled by $C_2$ by pre-installing wildcard rules on $V_2$ and $V_4$, but this may cause a hot spot on $C_1$. Furthermore, $f_3$ can be handled by $C_2$ depicted in Fig. 1(b) through pre-installing wildcard rules on $V_4$, but congestion may occur on the bottleneck link $(V_2, V_4)$ when some middlebox functions in VMs alter the flow sizes. Therefore, it's necessary to take both controller load balancing and link load balancing into account. The work [28] explored load balance in both control and data planes but they ignored the network policy.

Different from above works, we study both controller load balancing and link load balancing while taking the traffic rate effect of middleboxes into account. The key contributions are summarized as follows.

1) First, we formulate the load balance problem as a joint traffic-aware consolidated middleboxes selection and routing (JTMSR) problem and prove its NP-hardness. The traffic rate changing of middleboxes is considered when formulating JTMSR problem.
2) Then, we design a two-phase RL_RFRD (Flow Redirecting by Root Controller (R_FR) and Rounding-based Flow Directing by Local Decision Controller (L_RD)) algorithm to solve JTMSR where the first phase properly adopts wildcard rules to redirect selected flows and proposes R_FR algorithm while the second phase formulates a sub-problem of JTMSR called s-JTMSR and proposes a fast heuristic rounding-based algorithm L_RD to find fine-balanced middleboxes selection and routing path for each flow. The approximation performance of both R_FR and L_RD is respectively analyzed.
3) Finally, we evaluate the performance of the proposed algorithm in ITALYNET and Fat-tree topologies. Simulation results show that RL_RFRD has near-optimal load-balancing performance, and it can reduce average response time by about 9.7% compared with other existing algorithms.

The rest of this article is organized as follows. Section II surveys some related works. Then, system model and detailed problem formulation are described in Section III. In Section IV, we propose a two-phase algorithm to solve problem and analyze the approximate performance. Next, we evaluate the performance of the proposed algorithm in Section V. Finally, Section VI concludes this article and discusses the future work.

## II. Related Work

In this section, we will discuss the existing related works on TE in middlebox-supported physical networks and switch migration in multiple controllers.

### A. TE in Middlebox-Supported Physical Networks

SIMPLE [4] presented a SDN-based policy enforcement layer for middlebox specific traffic steering, but the number and position of middleboxes are fixed after network initialization, leading to inefficient resource utilization in traffic changing. So some works on SDN-based middlebox placement are emerged. Ma *et al.*, [13]–[15] formulated this problem

TABLE I
OBJECTIVE-BASED CLASSIFICATION OF MIDDLEBOX PLACEMENT AND ROUTING APPROACHES

| Objective | Load Balancing | Resource Usage | QoS-aware | | | Cost-aware | |
|---|---|---|---|---|---|---|---|
| | | | Aggregate Traffic | Latency | Throughput | Energy Cost | Operating Cost |
| Reference | [4], [8], [13]–[15] | [17], [20] | [9], [30] | [12], [16] | [10], [11], [31] | [19] | [18], [21], [32]–[34] |

and designed a placement solution, Liu *et al.*, [16] also studied this problem to improve service chaining performance and Bari *et al.*, [32] tried to optimize network operational cost where middlebox functions refer to VNFs and operational cost includes deploy cost, energy cost and cost of forwarding traffic. In [20], Kuo *et al.* proposed a chain deployment algorithm to follow the guidance of the link and server usage and explored each VNF in service chain whether to use additional server resources while in [21], Luizelli *et al.* proposed a deployment algorithm for service chains by minimizing the actual cost of virtual switching and mentioned the switching cost of a given service chain and [34] evaluates switching cost when performing service chain.

In addition, some works on joint VNF placement and routing are also studied with different objectives such as minimize the VNF deployment cost [33], maximize the amount of processed requests [30] and aggregate throughput [31]. Moreover, in [9] and [10], the authors addressed resource constraints in the network and proposed SDN-based routing with middleboxes. Furthermore, Alqarni *et al.*, [8] considered load balance of middleboxes with the objective of minimizing the communication energy cost of VM pairs, Huang *et al.*, [11] studied a joint optimization of middlebox selection and routing problem to maximize the throughput in SDN network and Duan *et al.*, [12] proposed a software-based middlebox scheme to minimize transmission latency. In addition, the authors in [17]–[19] aimed not only to determine the optimal location for placing VNFs, but also minimize the resource utilization with different objectives. Table I summarizes above works with different objectives.

There are also some pioneering works on virtual network embedding (VNE), but the problem of VNE is a little different from our work, where the problem refers to allocate efficient substrate resource to incoming network functions while the consolidated middleboxes in this article are placed at the fixed locations and the aim is to select middleboxes and find the routing path in multiple SDNs.

### *B. Switch Migration in Multiple Controllers*

Switch migration or dynamic assignment can avoid the controller load imbalance in multiple SDN controllers by permitting one switch to change its connected controller from the original one to the target one. In [25], Cheng *et al.* designed the maximum resource utilization migration algorithm and randomly selected a switch to migrate by overloaded controller, but this overloaded controller may be overloaded again after migration. Wang *et al.*, [26] pursued the efficiency of switch migration, but the controller response time will increase caused by selecting a reasonable migration pairing. In addition, Wang *et al.*, [24], [35] proposed a dynamic SDN controller assignment scheme in data center networks with the aim to balance the controller load while keeping the controller response time low. But these works ignored the overhead caused by switch migration. Xu *et al.*, [27] designed a dynamic switch migration method to achieve controller load balance with small migration cost. However, the cost of switch migration is actually high and frequent manipulation is not allowed [28].

Besides, some other researchers proposed to deploy default paths in advance to minimize controller response time [29] and designed an efficient optimization strategy associated with NFV to overcome resilience among multiple controllers [36], and He *et al.*, [37] designed dynamic controller plane by considering controller migration and switch reassignment and solved this problem in an online way with the advantage of simulated annealing technique. Motivated by [29], wildcard rules are chose and applied in this article to achieve both controller load balancing and link load balancing. The authors in [28] have also explored a new scheme to achieve both controller and link load balancing in SDN, but routing policy requirement is ignored when scheduling flows in this work.

## III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we will describe the system model, define joint traffic-aware consolidated middleboxes selection and routing (JTMSR) problem and prove its NP-Hardness. Commonly used variables and notations in this article are summarized in Table II.

### *A. System Model*

Our system model is built on the hierarchical control plane structure [38], which requires a global view of the network to achieve an optimal allocation of controller resource as shown in Fig. 1. The control plane includes a root controller (RC) and a set of local controllers, and each switch in data plane is connected to one master local controller (LC). RC will periodically determine the set of flows to be redirected and assign the wildcards to corresponding SDN-enabled switches based on the statistics information, e.g., link load and the number of flow entries, collected by each LC while each LC will dispatch user requests by allocating optimal routing path, installing forwarding rules into the forwarding tables in the switches, and assigning the middleboxes for the requests to PMs. Note that the proposed strategy in hierarchical control plane structure can also be employed in flat control plane [39] by applying a leader election algorithm to choose the central controller, which will be discussed in Section IV-C.

*1) Network Model and Resources:* In control plane, it consists of a RC $C_0$ and set of local controllers C associated with $C_0$. Each LC $c \in$ C has the processing capacity denoted as $\alpha_c$ in terms of the number of requests it can handle at one

TABLE II
LIST OF COMMONLY USED VARIABLES AND NOTATIONS

| Variables | Descriptions |
|---|---|
| C, V, E | Set of controllers, switches and links |
| $C_0$ | The root controller |
| $\alpha_c$ | Processing capacity of controller $c \in$ C |
| $V_c$, $V_p$ | Set of switches associated with controller $c$ and that directly connect to the PMs |
| $P_u$ | Capacity of PM located at switch $u$ |
| $B_{(u,v)}$ | Capacity of link $(u, v)$ |
| $T_u$ | The number of rules for flow directing on switch $u$ |
| $\mathcal{F}$ | Set of macroflows in the network |
| $s_f$, $t_f$ | Source switch and destination switch for flow $f$ |
| $d_f$ | Initial traffic demand at source switch for flow $f$ |
| $d_f^{u-}$, $d_f^{u+}$ | Traffic rate of flow $f$ before entering and leaving switch $u$ |
| $\mathcal{P}_f$ | Set of redirecting paths for flow $f$ from $s_f$ to $t_f$ |
| $n_f$ | The number of flow requests for flow $f$ |
| $m_f$ | Service chain for flow $f$ |
| $V_m$ | Set of switches that can perform $m_f$ |
| $\varepsilon_f$ | The ratio between the volumes of flow $f$ after and before processing by VM |
| $P_f^u$ | Computing resource consumption by performing $m_f$ on switch $u \in V_m$ for flow $f$ |
| $\tau_f$ | The number of routing rules for flow $f$ |
| $\lambda_u(t)$ | Request rate of switch $u$ in time slot $t$ |
| $\theta_c(t)$ | Load of controller $c$ in time slot $t$ |
| $x_f^{ul}$ | A binary variable whether switch $u \in V_m$ along path $l$ is selected to perform $m_f$ |
| $y^l$ | A binary variable whether path $l$ is selected for flow routing |
| $y^l_{(u,v)}$ | A binary variable whether link $(u, v)$ is selected by path $l$ |

time unit. From the view of data plane, the network topology can be modeled by $G = (\mathrm{V}, \mathrm{E})$, where V and E are the node set and edge set, respectively. Each node $u \in$ V denotes a SDN-enabled switch while each edge (*u*,*v*) $\in$ E represents a link from switch *u* to switch *v*. For each LC $c \in$ C, the set of its connected switches is denoted by $V^c$.

Let $V_p$ denote the set of switches that directly connect to the PMs and are able to implement middleboxes as VMs. Since each switch usually connects to PM through a high-speed optical link, the latency between them is negligible. We assume that the attached PM for a switch will be used interchangeably. Each switch $u \in$ V is equipped with a TCAM forwarding table that can accommodate at most $T_u$ rules for flow routing and let $\tau_f$ be the number of rules in the routing solution for flow *f*. Additionally, each PM attached to a switch $u \in V_p$ has limited capacity resource $P_u$. If switch $u \in V \setminus V_p$, then $P_u = 0$. Similarly, each link $(u, v) \in$ E has a bandwidth capacity $B_{(u,v)} \geq 0$, i.e., the available bandwidth.

*2) Network Workflow:* Following [14], [29], we concentrate on the persistent and large-size elephant flows in this article. This is because large size flows play a major role in the network traffic and it is beneficial to help balance the entire network by optimizing them efficiently while mice flows are transient and may leave network before the calculated optimization scheme. In the network, a set of macroflows is denoted by $\mathcal{F}$ and each macroflow $f \in \mathcal{F}$ is represented as a 4-tuple $(s_f, t_f, d_f, n_f)$, where $s_f$ and $t_f$ are source and destination switches, respectively, $d_f$ is the initial traffic rate at the ingress point and $n_f$ is the number of flow requests to be handled. Each flow *f* will be assigned a set of potentially redirecting paths $\mathcal{P}_f$ which are pre-calculated by root controller. A set of feasible paths $\mathcal{P}_f$ from $s_f$ to $t_f$ is defined as follows: if $|s_f v'| + |v' t_f| \leq (1 + \varpi)|s_f t_f|$, then the path *p*' from $s_f$ to $t_f$ via switch *v*' belongs to $\mathcal{P}_f$, i.e., $p' \in \mathcal{P}_f$, where $\varpi$ is predefined constant, such as 0.5, and $|s_f t_f|$ is the minimum hop number between two switches $s_f$ and $t_f$. Generally, these feasible paths which can be found by the shortest path algorithm are non-loop paths.

*3) Network Functions:* Virtualization enables implementation of the consolidated middleboxes and it assumes that all required network functions of each flow are obtained at a single PM and represented by a service chain $m_f$ which denotes a sequence of network functions that are chained together and has to be traversed in the specified order. Following [3], [4], [9], [10], [40], we assume that each middlebox function in $m_f$ is running at a single VM and different VMs serving different requests can be consolidated to a single PM. Specifically, when the flow *f* arrives at the PM hosting the VM for its service chain $m_f$, it will be directed to the VM and the functions in $m_f$ are applied in the specified order. Thus, performing the functions in $m_f$ will consume the computing network resource $P_f^u$ for $f$ of a PM attached to the switch $u \in V_p$. Let $V_m \subseteq V_p$ be the set of switches that can perform $m_f$.

Additionally, it is noted that some middlebox function may alter the traffic rate of flow on the link load [10], [13], [14]. In other words, traffic rate of flow increases if encryption is applied, while it decreases if compression is applied. We thereby define $\varepsilon_f > 0$ as the ratio between the traffic rate of *f* before and after processing by VM, where the value of $\varepsilon_f$ is given and can be derived from historical traffic [41]. We use $d_f^{u-}$ and $d_f^{u+}$ to denote the traffic rate of flow *f* before entering and leaving switch *u*, respectively. Note that $d_f^{s_f-} = d_f$. If service chain $m_f$ is processed by VM at switch *u* for short, then $d_f^{u+} = d_f^{u-} \cdot \varepsilon_f$. For convenience, we use $d_f^{(u,v)} = d_f^{u+} = d_f^{v-}$ to represent the traffic rate of *f* on its path link (*u*,*v*). Therefore, admission of flow *f* involves routing the traffic from $s_f$ to $t_f$ via a path $l \in \mathcal{P}_f$ subject to the constraints $d_f$ and $m_f$.

*4) Controller Load Model:* Following [24], [29], we consider a discrete time model where the length of time slot matches the timescale when switch requests can be precisely recorded. Each flow request is required to be sent to at least one controller, so the number of flows, that will be reported to the associated controller $c \in$ C by switch *u* in slot *t*, is denoted by $\lambda_u(t)$. Then, the load of controller *c* is

$$\theta_c(t) = \sum_{u \in V^c} \lambda_u(t) \tag{1}$$

Without confusion, we omit the parameter *t* in the variables $\lambda$ and $\theta_c$ in the following description.

## *B. Problem Formulation*

In order to formally define JTMSR problem, we will first introduce some variables, and then give the objectives and constraints.

Generally, middlebox functions mapped on the switches that directly connected to PMs are given in advance, and the same type of middleboxes are deployed with multiple copies at various locations. This scenario is practical in reality [11]. We define a binary variable $x_f^{ul}$ as follows to indicate whether the switch *u* along path *l* that is able to perform $m_f$ for flow *f* is selected,

$$x_f^{ul} = \begin{cases} 1, & \text{if } m_f \text{ is running on switch } u \in V_m \\ & \text{along path } l. \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Generally, the optimal routing path *l* for each flow is assigned by controller. However, to reduce the controller response time, the wildcards are wildly used for large size flows, and potentially flow redirecting path set $\mathcal{P}_f$ can be pre-calculated [29]. Thus, the path *l* for each flow generally consists of two parts: flow redirecting path and flow directing path. For uniform description, a special path $\phi_f$ is added to set $\mathcal{P}^f$, in order to denote whether this macroflow is redirected. If $\phi_f$ is selected, it means that this flow will not be redirected. In this case, the $\phi_f$ is the flow directing path for flow *f*. Therefore, we define a binary variable $y^l$ for routing path selection as follows:

$$y^l = \begin{cases} 1, & \text{if path } l \text{ is selected for flow routing.} \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

For each flow *f*, it will be routed through one potential path by controller at most, which is denoted as

$$\sum_{l \in \mathcal{P}_f} y^l = 1 \quad \forall f \in \mathcal{F} \quad (4)$$

Then, the number of processing flows that will be reported to each LC by switch $u \in \mathrm{V}$ is represented by

$$\lambda_u = \sum_{u = last(l): l \in \mathcal{P}_f} y^l \cdot n_f \quad \forall u \in \mathrm{V} \quad \forall f \in \mathcal{F} \quad (5)$$

where *last*(*l*) denotes the last switch of path *l*, and the number of processing flows on each LC cannot exceed the capacity, which is expressed as

$$\theta_c = \sum_{u \in V_c} \lambda_u \le \alpha_c \quad \forall c \in \mathrm{C} \quad (6)$$

Thus, the maximum load difference ratio $\eta_1$ among controllers is calculated by

$$\eta_1 = \frac{|\mathrm{C}|}{\sum_{c \in \mathrm{C}} \theta_c} (\max \theta_c - \min \theta_c) \quad \forall c \in \mathrm{C} \quad (7)$$

Note that the max and min values indicate the maximum and minimum load of controller. They are calculated and identified by root controller (RC) and not linearized to LP version. LP version is used to solve link load balancing in the data plane via L_RD algorithm and the controller load balancing is solved by R_FR algorithm.

With respect to service chain $m_f$, it will be processed by at least one switch along this path *l*, which is denoted as

$$\sum_{u \in V_m} x_f^{ul} \ge y^l \quad \forall l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \quad (8)$$

Besides, if middlebox function is running on VM placed at switch *u*, the traffic rate on a link *(u,v)* is denoted as

$$d_f^{(u,v)} = d_f^{u-} \cdot \varepsilon_f \cdot x_f^{ul} \quad \forall (u,v) \in \mathrm{E} \quad l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \quad (9)$$

and available processing capacity of each middlebox should not be exceeded, which is denoted as

$$\sum_{f \in \mathcal{F}} \sum_{u \in l: l \in \mathcal{P}_f} x_f^{ul} \cdot P_f^u \le P_u \quad \forall u \in V_p \quad (10)$$

Additionally, if a path $l \in \mathcal{P}_f$ is selected, the number of installed rules on each switch $u \in \mathrm{V}$ should not exceed $T_u$

$$\sum_{f \in \mathcal{F}} \sum_{u \in l: l \in \mathcal{P}_f} y^l \cdot \tau_f \le T_u \quad \forall u \in \mathrm{V} \quad (11)$$

Thus, for a link *(u,v)*, its load ratio should be less than or equal to the optimization objective $\eta_2$.

$$\frac{\sum_{f \in \mathcal{F}} \sum_{(u,v) \in l: l \in \mathcal{P}_f} y^l d_f^{(u,v)}}{B_{(u,v)}} \le \eta_2 \le 1 \quad \forall (u,v) \in \mathrm{E} \quad (12)$$

The goal of JTMSR problem is to minimize maximum link load ratio while balancing the load among controllers. Hence, we apply a weight factor $\omega \in [0,1]$ in the objective function. Mathematically, we have the following formulation.

$$\begin{aligned} \min \quad & \omega\eta_1 + (1-\omega)\eta_2 \\ \text{s.t.:} \quad & (4) - (12) \\ & x_f^{ul}, y^l \in \{0,1\} \end{aligned} \quad (13)$$

$\eta_1$ and $\eta_2$ respectively indicate the maximum load difference ratio among controllers and link load ratio. Note that different from the existing works on middlebox scheduling, on the one hand, the model considers that most middleboxes are traffic processing devices, which may significantly change the volume of processed flows. On the other hand, it considers the scenario of multiple distributed SDNs to address the limitation of a single controller which has limited processing capacity and easily becomes bottleneck of SDN.

## C. NP-Hardness

Before proving the NP-hardness of JTMSR, we first give the definition of Identical Parallel Machines Scheduling Problem (IPMSP), which has been proved to be NP-hard [42] and the reduction is based on the NP-complete Partition problem [43].

*Definition 1 (IPMSP):* Given *m* parallel machines, *q* independent jobs, each job with a processing time $u_i$ on each machine with $1 \le i \le q$, find a schedule where each job is assigned to one of the machines so as to minimize the makespan.

*Theorem 1:* The JTMSR problem is NP-hard.

*Proof:* In order to prove that JTMSR problem is a NP-hard problem, we consider a simple example of JTMSR problem. In the control plane, there are one root controller and two local controllers while in the view of data plane, network topology can be modeled by $G = (V, E)$. There are *q* flows to be

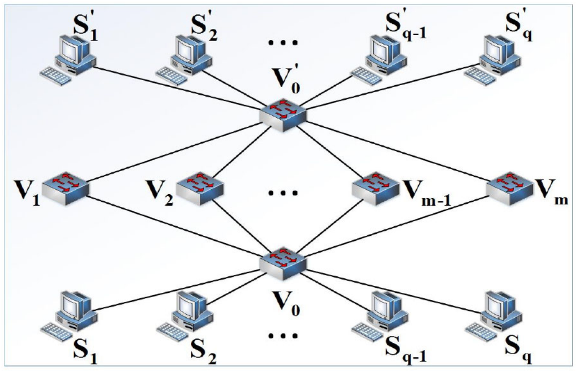


Fig. 2. A special example of the JTMSR problem.

directed in the network shown in Fig. 2, and the capacities of links $(V_0, V_i)$ and $(V_i, V_0')$ with $1 \leq i \leq m$ are set as $B_{(V_0,V_i)}$ and $\infty$, respectively. After determining the LC for handling each flow from $S_j$ to $S_j'$ with $1 \leq j \leq q$, we only need to choose another feasible path except pre-calculated path as default path where wildcard rules are assigned to available switches. To simplify the problem, we assume that network computing resources in $V_p$ are enough and middlebox function of service chain for each flow is performed correctly, then we will choose one remaining feasible path for each flow to forward its traffic amount, $u_i \cdot B_{(V_0,V_i)}$, so as to minimize maximum traffic link load ratio in a network.

By regarding each flow from $S_j$ to $S_j'$ and each link $(V_0, V_i)$ as a job *j* and a machine *i*, the processing time for each job *j* is $\frac{u_i \cdot B_{(V_0,V_i)}}{B_{(V_0,V_i)}} = u_i$, so this is an instant of IPMSP problem. If the paths are determined for transmitting flows, then the link load ratio can be calculated in the polynomial time. Therefore, the JTMSR problem can be reduced to the IPMSP problem in the polynomial time. Since IPMSP problem is NP-hard problem, our JTMSR problem is also NP-hard. This completes proof. ■

## IV. Algorithm Design

Due to the NP-hardness of JTMSR problem, a heuristic algorithm with polynomial time is designed and sub-optimal solution is obtained. In this section, we propose a two-phase heuristic algorithm RL_RFRD which consists of two parts, flow redirecting (R_FR) by root controller and flow directing (L_RD) by local decision controller (LDC). Particularly, we will first introduce R_FR and L_RD in detail and analyze the approximate performance respectively, and then give some discussions on our proposed algorithm.

### *A. Flow Redirecting by Root Controller*

RC is responsible for assigning the wildcard rules to redirect macroflows in advance to achieve load balance of local controllers and reduce the controller response time, and then it determines a LC with least controller load as LDC to dispatch these flow requests to reduce the link congestion in the data plane. RC will send the state information of other local controllers and flow redirecting path to LDC. The traffic

```
Algorithm 1 Flow Redirecting by Root Controller (R_FR)
Require: Controllers $C_0$ and C in control plane, network
    topology $G = (V, E)$ in data plane, set of flows $\mathcal{F}$ and
    set of feasible paths $\mathcal{P}_f$ for each flow $f \in \mathcal{F}$.
Ensure: LDC $C_f^{\min}$ and the switch $v \in V^{C_f^{\min}}$ that reports
    flow requests to $C_f^{\min}$.
1: Generate the set of local candidate controllers for flow $f$
    denoted as $C_f^{cad}$ according to feasible paths by $C_f^{cad} \leftarrow \{c | u \in V^c, \forall u \in l{:}l \in \mathcal{P}_f\}$
2: Select the controller with minimum controller load as
    $C_f^{\min}$ among $C_f^{cad}$ where $C_f^{\min} \leftarrow \min_{c \in C_f^{cad}} \{\theta_c\}$
3: Select the switch $v$ in $V^{C_f^{\min}}$ that will report
    the flow request of flow $f$ to LDC by $v \leftarrow \arg\min_{(u,v) \in \mathcal{P}_f} L_{(u,v)}, \forall u \in \mathrm{V} \backslash V^{C_f^{\min}}, v \in V^{C_f^{\min}}$
4: Judge whether there is a switch on current redirected path
    satisfying middlebox capacity constraint
5: Update network information including controller capacity,
    link capacity and middlebox capacity
```

sizes in the current time slot can be predicted through some prediction schemes, which will be discussed in Section IV-C. Take Fig. 1(c) as an example, to balance the load of controllers, $f_3$ from $V_3$ to $V_6$ will first be redirected by switch $V_5$, and LC $C_2$ will be chose as the LDC to handle this flow. Algorithm 1 describes this process.

In Algorithm 1, RC $C_0$ first generates the set of local candidate controllers as $C_f^{cad}$ according to feasible paths of flow *f* and then tries to find out the LDC with minimum controller load as $C_f^{\min}$ among $C_f^{cad}$ shown in lines 1-2. Then LDC will handle the flow request of *f* by assigning routing rules to the corresponding switches described in IV-B. In Fig. 1(c), LDC $C_2$ will assign feasible path $V_5 \rightarrow V_4 \rightarrow V_6$ to complete routing for $f_3$ to minimize the link load under the link bandwidth capacity, switch memory and middlebox capacity constraints. Then, in line 3, RC will determine the switch $v \in V^{C_f^{\min}}$ on link $(u, v)$ with the least link load to report flow requests to $C_f^{\min}$, where $L_{(u,v)}$ is the link load of $(u, v)$. Since wildcards are installed in advance, the flow *f* can be directly redirected to switch *v*. Next, judge whether there is a switch on current redirected path satisfying middlebox capacity constraint in line 4. If this is no corresponding switch, it is required to find one switch in the remaining path to meet this constraint, otherwise, this flow will be dropped. Finally, in line 5, update the network information both in control and data planes, including controller capacity, link capacity and middlebox capacity.

Thus, for a flow in flow set $\mathcal{F}$, the computing complexity of Algorithm 1 is $O(k^2)$ at the worst where *k* is the number of local candidate controllers in the network and $k \leq |\mathrm{C}|$. Then, it can be proved that load difference among local candidate controllers can be minimized after choosing the controller whose load is minimum.

*Theorem 2:* For each flow *f*, suppose that there is *k* local candidate controllers in $C_f^{cad}$ and the current load of controllers in the network is $\theta_{C_1^{cad}} \geq \theta_{C_2^{cad}} \geq \cdots \geq \theta_{C_j^{cad}} \cdots \geq$

$\theta_{C_i^{cad}} \cdots \geq \theta_{C_k^{cad}}$ with $1 < j \leq i < k$, the current maximum load difference $\eta_2$ among controllers depends on $\theta_{C_1^{cad}} - \theta_{C_k^{cad}}$ according to Eq. (7). After choosing $C_k^{cad}$ to handle flow requests $n_f$, the load difference among controllers can be totally minimized.

*Proof:* Suppose the controllers whose load more than $\theta_{C_j^{cad}}$ cannot handle the flow request of *f*, i.e., $\theta_{C_{j-1}^{cad}} + n_f \geq \alpha_{C_{j-1}^{cad}}$, so in this case, we should choose controllers whose capacity is enough to handle this flow request. These controllers can be divided two categories, the controller whose current load is minimum and rest of controllers except overloaded controller, i.e., $C_k^{cad}$ and $C_i^{cad}$ with $j \leq i < k$. Let $\hat{\eta}_2 = \theta_{C_1^{cad}} - \theta_{C_k^{cad}}$. Then, we can compare the load difference with $\hat{\eta}_2$ after choosing $C_i^{cad}$ and $C_k^{cad}$, respectively.

Case 1: For $C_i^{cad}$ with $j \leq i < k$ being chose, the minimum load difference among controllers $\triangle\eta_{C_i^{cad}}$ is

$$\triangle\eta_{C_i^{cad}} = \begin{cases} \theta_{C_i^{cad}} + n_f - \theta_{C_k^{cad}}, & \text{if } \theta_{C_i^{cad}} + n_f \geq \theta_{C_1^{cad}}, \\ \theta_{C_1^{cad}} - \theta_{C_k^{cad}}, & \text{otherwise.} \end{cases}$$

It is obvious that $\triangle\eta_{C_i^{cad}} \geq \hat{\eta}_2$ when choosing $C_i^{cad}$ with $j \leq i < k$.

Case 2: For $C_k^{cad}$ being chose, let $i = k - 1$, then the minimum load difference among controllers $\triangle\eta_{C_k^{cad}}$ is

$$\triangle\eta_{C_k^{cad}} = \begin{cases} \theta_{C_k^{cad}} + n_f - \theta_{C_i^{cad}}, & \text{if } \theta_{C_k^{cad}} + n_f \geq \theta_{C_1^{cad}}, \\ \theta_{C_1^{cad}} - (n_f + \theta_{C_k^{cad}}), & \text{if} \theta_{C_k^{cad}} + n_f \leq \theta_{C_i^{cad}}, \\ \theta_{C_1^{cad}} - \theta_{C_i^{cad}}, & \text{otherwise.} \end{cases}$$

Obviously, $\triangle\eta_{C_k^{cad}} \leq \hat{\eta}_2$ when choosing $C_k^{cad}$ with $i = k-1$.

Therefore, it is best to choose the local candidate controller whose current load is the minimum. ■

### *B. Flow Directing by Local Decision Controller*

Shortest path scheme widely adopted to find the routing path for each flow is not suitable and high congestion may occur in specific bottleneck links if middlebxoe selection and traffic routing are not well jointly planned [11]. As shown in Fig. 1(b), the routing path for flow $f_3$ is $V_3 \rightarrow V_2 \rightarrow V_4 \rightarrow V_6$ which balances the controller load but the link congestion cannot be avoided while in Fig. 1(c), the path is $V_3 \rightarrow V_2 \rightarrow V_5 \rightarrow V_4 \rightarrow V_6$ which balances the controller load and avoids link congestion simultaneously. Therefore, after obtaining LDC to handle flow request *f* and switch *v* to report flow request by Algorithm 1, LDC will search the remaining fine-grained paths from *v* to $t_f$ to route the flow demand $d_f$ so as to minimize the maximum link load ratio under the constraints of link capacity, switch memory and middlebox capacity. This process can be formulated as a sub-problem of JTMSR called s-JTMSR.

*1) s-JTMSR Problem Formulation:* For many flow requests in practice, their fine-grained candidate paths are generally not specified, i.e., the forwarding rules are decided by LDC. Therefore, we define a binary variable $y^l_{(u,v)}$ to indicate whether link (*u*, *v*) is selected by path *l*,

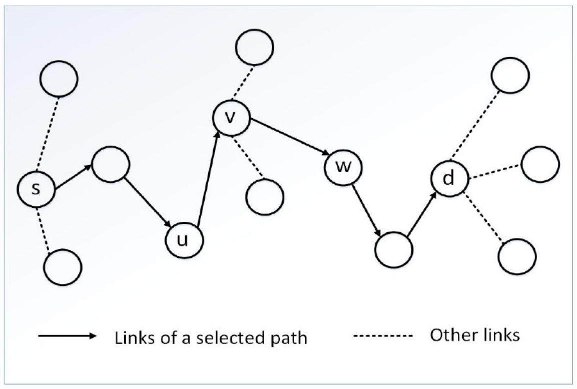


Fig. 3. An example of path searching.

$$y^l_{(u,v)} = \begin{cases} 1, & \text{if link } (u,v) \text{ is on the path } l. \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

The path searching process is represented by constraints (15)-(17). For $\forall l \in \mathcal{P}_f$ $\forall f \in \mathcal{F}$,

$$\sum_{(s_f,u)\in \mathrm{E}} y^l_{(s_f,u)} - \sum_{(u,s_f)\in \mathrm{E}} y^l_{(u,s_f)} = 1 \quad (15)$$

$$\sum_{(u,t_f)\in \mathrm{E}} y^l_{(u,t_f)} - \sum_{(t_f,u)\in \mathrm{E}} y^l_{(t_f,u)} = 1 \quad (16)$$

$$\sum_{(u,v)\in \mathrm{E}} y^l_{(u,v)} - \sum_{(v,w)\in \mathrm{E}} y^l_{(v,w)} = 0 \quad \forall v \in \mathrm{V}\backslash\{s_f, t_f\} \quad (17)$$

Specifically, constraints (15) and (16) enforce that each flow *f* starts and ends at its source node $s_f$ and destination node $t_f$, respectively. Constraint (17) guarantees flow conservation at each intermediate node on the path, i.e., the number of incoming links should be equal to the number of outgoing links. Take Fig. 3 as an example, a path is shown in solid arrow from source node *s* to destination node *d*. If a path is selected, for *s*, the number of outgoing links minus that of incoming links equals to 1, otherwise, their differences should be 0, and the same constraint is imposed for *d*. As for each intermediate node such as *v*, the number of incoming links should be equal to that of outgoing links.

Constraints (18) and (19) impose to avoid loop paths, where $z^l_{(u,v)}$ is an integer variable to indicate the sequence number of the link along path *l*, i.e., (*u*,*v*) is the $z^l_{(u,v)}$-th link along the path *l*.

$$0 \leq z^l_{(u,v)} \leq y^l_{(u,v)} \cdot (|\mathrm{V}| - 1) \quad \forall (u,v) \in l \quad \forall l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \quad (18)$$

$$\sum_{(v,w)\in \mathrm{E}} z^l_{(v,w)} - \sum_{(u,v)\in \mathrm{E}} z^l_{(u,v)} = \sum_{(v,w)\in \mathrm{E}} y^l_{(v,w)} \quad \forall v \in \mathrm{V}\backslash\{t_f\} \quad \forall l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \quad (19)$$

Particularly, if the link (*u*,*v*) is not on path *l*, i.e., $y^l_{(u,v)} = 0$, the value of $z^l_{(u,v)}$ should be 0. Otherwise, the difference between the sequence numbers of two consecutive links on the

**Algorithm 2** Rounding-Based Flow Directing by Local Decision Controller (L_RD)

```
Require: Problem formulations with integer variables $y^l_{(u,v)}$, $x^{ul}_f \in \{0,1\}$
Ensure: Solutions $\tilde{y}^l_{(u,v)}$, $\tilde{x}^{ul}_f$ of the original problem.
1: Obtain the solutions $\hat{y}^l_{(u,v)}$ and $\hat{x}^{ul}_f$ of optimization problems by relaxing all integer variables
2: for all $f \in \mathcal{F}$ do
3:     for all $l \in \mathcal{P}_f$ do
4:         $\tilde{y}^l_{(u,v)} \leftarrow$ RoutingPathSearch ($\hat{y}^l_{(u,v)}, f, l$)
5:         $\tilde{x}^{ul}_f \leftarrow$ MFuncSelection ($\hat{x}^{ul}_f, \tilde{y}^{ul}_f, f, l$)
6:     end for
7: end for
```

path should be 1. Therefore, if link (*u,v*) is on path *l*, $z^l_{(u,v)}$ is between 0 and $|\mathrm{V}| - 1$.

With respect to service chain, it always shall be guaranteed that middlebox function should be processed on and only on the switches along the selected paths, which is shown in constraints (20)-(21)

$$x^{ul}_f \le \sum_{(u,v)\in E} y^l_{(u,v)} + \sum_{(v,t_f)\in E} y^l_{(v,t_f)} \quad \forall u \in V_m \quad \forall l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \tag{20}$$

$$\sum_{u\in V_m} x^{ul}_f \ge \sum_{(s_f,v)\in E} y^l_{(s_f,v)} \quad \forall l \in \mathcal{P}_f \quad \forall f \in \mathcal{F} \tag{21}$$

Additionally, if a path $l \in \mathcal{P}_f$ is selected, the number of installed rules on each switch $u \in \mathrm{V}$ should not exceed $T_u$

$$\sum_{f\in\mathcal{F}} \sum_{u\in l:l\in\mathcal{P}_f} y^l_{(u,v)} \cdot \tau_f \le T_u \quad \forall u \in \mathrm{V} \tag{22}$$

Similarly, available processing capacity of each middleboxes should not be exceeded, which is given by

$$\sum_{f\in\mathcal{F}} \sum_{u\in l:l\in\mathcal{P}_f} x^{ul}_f \cdot P^u_f \le P_u \quad \forall u \in V_p \tag{23}$$

And for a link (*u, v*), its link load ratio should be less than or equal to the optimization objective $\eta_2$. Thus, the constraint (12) is rewritten as (24).

$$\frac{\sum_{f\in\mathcal{F}} \sum_{l\in\mathcal{P}_f} y^l_{(u,v)} d^{(u,v)}_f}{B_{(u,v)}} \le \eta_2 \le 1 \quad \forall (u,v) \in \mathrm{E} \tag{24}$$

Thus, s-JTMSR can be formulated as follows.

$$\begin{aligned} \min \quad & \eta_2 \\ \text{s.t.:} \quad & (9) \text{ and } (15)-(24) \\ & y^l_{(u,v)}, x^{ul}_f \in \{0,1\} \end{aligned} \tag{25}$$

The s-JTMSR problem is also NP-hard because each IPMSP instant is a special case of s-JTMSR. Because of the NP-hardness of the s-JTMSR, we propose a fast heuristic algorithm L_RD shown in Algorithm 2 by using relaxation and rounding techniques to solve the problem in (25).

**Algorithm 3** RoutingPathSearch

```
Require: LP solution $\hat{y}^l_{(u,v)} (\forall (u,v) \in \mathrm{E})$, flow index $f$ and path index $l$
Ensure: The rounded solutions $\tilde{y}^l_{(u,v)}$
1: $\tilde{y}^l_{(u,v)} \leftarrow 0, \forall (u,v) \in \mathrm{E}$
2: Sort $\Omega = \left\{(u,v) | \hat{y}^l_{(u,v)} > 0\right\}$ as $\chi_1, ..., \chi_{|\Omega|}$ such that $\hat{y}^l_{\chi_1} \ge ... \ge \hat{y}^l_{\chi_{|\Omega|}}$
3: $j = 1,\ \Psi \leftarrow \emptyset$
4: while $j \le |\Omega|$ do
5:     $\Psi \leftarrow \Psi \cup \{\chi_j\}$
6:     if $\lceil \hat{y}^l_{(u,v)} \rceil, \forall (u,v) \in \Psi$ satisfy (15), (16) and (17) then
7:         $\tilde{y}^l_{(u,v)} \leftarrow 1, \forall (u,v) \in \Psi$
8:         break
9:     end if
10:    $j = j + 1$
11: end while
```

**Algorithm 4** MFuncSelection

```
Require: LP solution $\hat{x}^{ul}_f$, $\tilde{y}^l_{(u,v)}$ from Algorithm 3, $f$ and $l$
Ensure: The rounded solutions $\tilde{x}^{ul}_f$
1: $\tilde{x}^{ul}_f \leftarrow 0, \forall u \in V_m, \forall f \in \mathcal{F}$
2: Sort $\Omega = \left\{(u, m_f) | \hat{x}^{ul}_f > 0\right\}$ as $\chi_1, ..., \chi_{|\Omega|}$ in a decreasing order of $\hat{x}^{ul}_f$
3: $\Lambda \leftarrow \left\{(u,v) | \tilde{y}^l_{(u,v)} = 1, \forall (u,v) \in \mathrm{E}\right\}$
4: $j = 1,\ \Psi \leftarrow \emptyset$
5: while $j \le |\Omega|$ do
6:     $(u', m_f{}') \leftarrow \chi_j$
7:     if $u' \in \Lambda$ and $\forall (u, m_f) \in \Psi, m_f{}' \ne m_f$ and $\sum_{\forall (u',m_f)\in\Psi} P^{u'}_{m_f} + P^{u'}_{m_f{}'} \le P_{u'}$ and $\sum_{\forall (u',m_f)\in\Psi} \tau_f + \tau_{f'} \le T_{u'}$ then
8:         $\Psi \leftarrow \Psi \cup \{\chi_j\}$
9:         if $V_m \subseteq \cup_{\forall (u,m_f)\in\Psi} \{m_f\}$ then
10:            $\tilde{x}^{ul}_f \leftarrow 1, \forall (u, m_f) \in \Psi$
11:            break
12:        end if
13:    end if
14:    $j = j + 1$
15: end while
```

2) *Rounding-Based Flow Directing by Local Decision Controller (L_RD):* In Algorithm 2, we first solve the optimization problem by relaxing all integer variables, and then obtain feasible solutions by invoking RoutingPathSearch shown in Algorithm 3 and MFuncSelection algorithms shown in Algorithm 4. In Algorithm 3, all (*u, v*) links are sorted in a decreasing order according to values of $\hat{y}^l_{(u,v)}$ and they are maintained in set $\Omega$. Then, the feasible solutions satisfying constraints (15)-(17) will be found in the **while** loop from lines 4 to 11.

Similarly, to find feasible solutions of $x^{ul}_f$ in Algorithm 4, we first sort the values of $\hat{x}^{ul}_f$ in a decreasing order and maintain feasible solutions in set $\Omega$. Then, all nodes belonging to the routes obtained from Algorithm 3 are maintained in set $\Lambda$. Next, each element $\chi_j = (u', m_f{}')$ in $\Omega$ is sequentially checked, and will be included in $\Psi$ if it satisfies the following conditions: i) node *u*' is in $\Lambda$, ii) service function $m_f{}'$ does not show in a $(u, m_f)$-tuple in $\Psi$, and iii) the remaining space on node *u*' can accommodate allocated rules $\tau_f$ as well as capacity of $m_f{}'$ in the **while** loop from lines 5 to 15.

*Theorem 3:* The time and space complexity of Algorithm 2 is $O(k^2 + |E| + |V| \cdot \sum_{f\in\mathcal{F}} |V_m||\mathcal{P}_f|)$ and $O(|E|\sum_{f\in\mathcal{F}}|\mathcal{P}_f| + |V|\sum_{f\in\mathcal{F}}|V_m||\mathcal{P}_f|)$, respectively.

*Proof:* The worst computational complexity of Algorithm 3 and 4 is $O(|\hat{y}^l_{(u,v)}|) = O(|E|)$ and $O(|\hat{x}^{ul}_f||\Lambda|) = O(|V||V_m||\mathcal{P}_f|)$, respectively. Therefore, the overall time complexity can be derived as follows.

$$O(Alg.\ 2) = O\left(\sum_{f\in\mathcal{F}}|\mathcal{P}_f| \times (O(Alg.\ 3) + O(Alg.\ 4))\right) = O\left(|E| + |V| \cdot \sum_{f\in\mathcal{F}}|V_m||\mathcal{P}_f|\right)$$

The space complexity is determined by the number of variables $y^l_{(u,v)}$ and $x^{ul}_f$ where $y^l_{(u,v)}$ is defined with respect to the number of paths and links and $x^{ul}_f$ is defined with respect to the number of paths and switches that can perform service chain. In the model, they are respectively calculated as $|E|\sum_{f\in\mathcal{F}}|\mathcal{P}_f|$ and $|V|\sum_{f\in\mathcal{F}}|V_m||\mathcal{P}_f|$. Since the set of switches *V*, the set of switches that perform service chain $V_m$ and the set of links *E* in the network are determined, variables $y^l_{(u,v)}$ and $x^{ul}_f$ are linear with the number of paths. Therefore, the model has a linear number of variables. In addition, by summing up the number of all these variables and the corresponding rounded ones, the total space is $O(|E|\sum_{f\in\mathcal{F}}|\mathcal{P}_f| + |V|\sum_{f\in\mathcal{F}}|V_m||\mathcal{P}_f|)$. ■

*3) Approximate Performance Analysis:* Assume that the minimum link capacity of all links is $B^{\min}_{(u,v)}$. We define three constant values as follows:

$$\alpha_0 = \min\left\{\frac{B^{\min}_{(u,v)}}{d^{(u,v)}_f}, f \in F\right\}$$
$$\alpha_1 = \min\{T_u, u \in \mathrm{V}\}$$
$$\alpha_2 = \min\{P_u, u \in V_p\} \quad (26)$$

Under many practical cases, $\alpha_0 \geq 1$ because the size of each flow is usually less than the link capacity. Besides, $T_u$ and $P_u$ are much larger than 1. Thus, it is reasonable to assume that $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ [44]. In the following, we give the approximation performance of the proposed L_RD algorithm.

*Lemma 1 (Link Capacity Constraint):* The proposed L_RD algorithm guarantees that the total traffic on any link (*u,v*) will not exceed the traffic load on the fractional solution by a factor of $\frac{4\log|\mathrm{V}|}{\alpha_0} + 3$.
The proof of Lemma 1 has been relegated to Appendix A.

*Lemma 2 (Flow-table Size Constraint):* After the rounding process in the L_RD algorithm, the number of required flow entries on any switch *u* will not exceed the constraint $T_u$ by a factor of $\frac{3\log|\mathrm{V}|}{\alpha_1} + 3$.
The proof of Lemma 2 has been relegated to Appendix B.

*Lemma 3 (Middlebox Capacity Constraint):* After the rounding process in the L_RD algorithm, the capacity required on any switch $u \in V_p$ will not exceed the middlebox capacity constraint $P_u$ by a factor of $\frac{3\log|V_p|}{\alpha_2} + 3$.
The proof of Lemma 3 has been relegated to Appendix C.

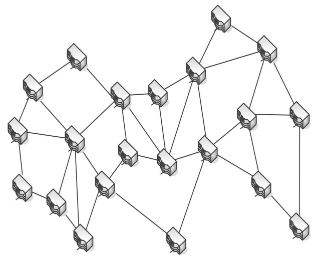
*Approximation Factor:* According to the above analysis, we get conclusion as follows.

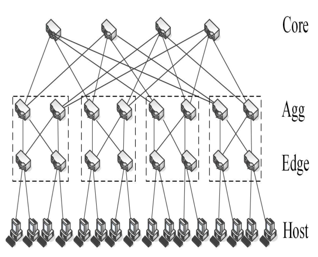
*Theorem 4:* The L_RD algorithm can guarantee that, the link capacity will hardly be violated by a factor of $\frac{4\log|\mathrm{V}|}{\alpha_0} + 3$, the flow-table size constraint will not be violated by a factor of $\frac{3\log|\mathrm{V}|}{\alpha_1} + 3$, and the middlebox capacity constraint will not be violated by a factor of $\frac{3\log|V_p|}{\alpha_2} + 3$ for s-JTMSR problem.

In most practical situations, the L_RD algorithm can reach almost the constant approximation in terms of link capacity constraint, flow-table size constraint and middlebox capacity constraint. For example, the link capacity of today's network will be 10Gbps and the maximum size of flow may reach 10Mbps by observing the practical traffic traces. Under two cases, $\frac{B^{\min}_{(u,v)}}{d^{(u,v)}_f}$ will be $10^3$ and the value of $\alpha_0$ will be $10^3$. Considering a network with 1000 switches, then we have $\log|\mathrm{V}| = 10$. Based on the Lemma 1, the approximation factor for the link capacity is 3.04. This also fits for the flow-table size constraint and middlebox capacity constraint. In other words, the L_RD algorithm can achieve the constant tri-criteria approximations for the s-JTMSR problem in many situations.

## *C. Discussion*

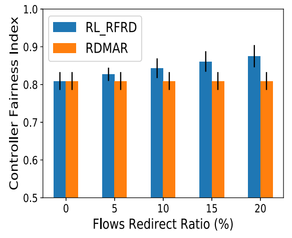
1) The proposed algorithm can guarantee the correctness of original optimization problem. Mathematically, Eq. (13) which consists of two parts coordinated by parameter $\omega$ can be expressed as $\min \omega\eta_1 + \min(1-\omega)\eta_2$. Technically, in hierarchical SDN-based network, selecting inter-domain routing path divides two steps, i.e., find the LDC by RC and calculate routing path by LDC. Thus, by ignoring the parameter $\omega$, it is reasonable in the algorithm design to solve two sub-problems and convert it into $\min \eta_1$ and $\min \eta_2$ for the sake of simplicity.
2) Additionally, this article aims to select paths based on the assumption that the switches are working normally. In fact, node in the SDN-based network that fails before path selection will not be selected in routing path and the node that fails during path selection may involve other issues, such as consistent flow forwarding rules updates [45] and failure detection and recovery mechanisms [46], which are beyond the scope of this article.
3) In some practical scenarios, the flow size of each flow is unknown before arriving. However, the flow size of each new-arrival in JTMSR is assumed to be known when RC assigns wildcard rules for these flows. It is reasonable to assume that the size and distribution of flows in the network can be predicted with bounded error [29] and


(a) ITALYNET

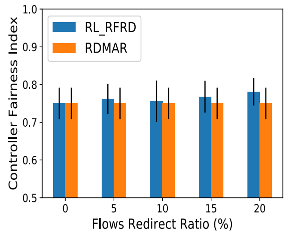


(b) 4-pod Fat-tree

Fig. 4. Simulation network structure.



(a) ITALYNET



(b) Fat-tree

Fig. 5. Controller fairness index vs. Flows redirect ratio.

many previous works [47]–[49] addressed the issue of flow size prediction can be applied in our system.

4) What's more, in the flat control plane, a RC can be found by applying leader election algorithm, which makes decision on installing wildcards for flow redirecting and finding the LDC for each flow. The effect of hierarchical control plane structure on performance is mainly reflected in the communication delay between RC and local controllers, which further affects the response time for each flow, but it has no effect on performance of data plane.

## V. Performance Evaluation

This section will discuss the experiment setup and evaluate the performance of proposed algorithms.

### A. Simulation Setup

We run all algorithms on a PC running on a Windows 64-bit with 12GB memory and a single Inter i5-7200U CPU using Python 2.7. All algorithms use a single processor core. ITALYNET topology [50] and **Fat-tree** topology [51], which is an architecture widely used in data centers, are adopted to measure the effectiveness of proposed algorithms. There are one root controller and four local controllers in the control plane and the network are divided into four domains. In ITALYNET, it consists of 21 switches and 10 additional middleboxes in the data plane. The processing capacity of each middlebox and the link capacity are all set to 150Mbps, and the processing capacity of controller is set to 1600 flows/s. In the experiment, the flows from the same source to the same destination switch are aggregated as a macroflow to route at each time slot. For each timeslot, there are [8, 20] macroflows and the traffic rate of each flow is generated within the range [12, 30] Mbps.

In Fat-tree, the number of pod is 8 with total 80 switches and 128 hosts. There are 20 additional middleboxes. The traffic processing capacity of each middlebox is set to 2000 Mbps. Besides, the link capacities between two connected switches are uniformly set to 2 Gbps and the processing capacity of each controller is 18K flows/s. At each time slot, the number of macroflows is set with the range of [20, 40] and the number of aggregated small flows is set with the range of [100, 300]. The request arrival rate follows the flow inter-arrival time distribution measured in real-world data center [52]. In both topologies, the number of types of middleboxes are set to 4 and each PM associates with middlebox functions ranging from 1 to 4. When redirecting flows, the shortest path is utilized by default and each selected flow is provided with candidate paths ranging from 3 to 10.

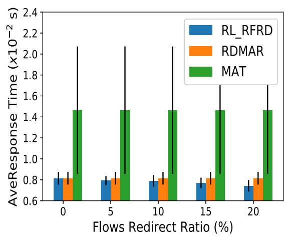
We compare our algorithm with the following schemes, i) **RL_RFRD** where flow redirecting is implemented by RC and the remaining path is determined by local controller; ii) **MAT** where the objective of formulation in [11] is adjusted to load factor and candidate routing paths are provided by a single controller; iii) **RDMAR** [28] where the static scheme between switches and multiple controllers is used without redirecting flows; iv) **OPT** where multiple SDNs are considered and routing path is solved by optimizer. For controller load balancing, RDMAR is the baseline where the static scheme is used while the OPT and MAT are the baselines for link load balancing. The flows for each timeslot are handled at the same time in multiple SDNs. All mathematical programming formulations are solved by Gurobi optimizer [53] and an example of Gurobi model is available.[1] The experiment carries out 30 runs for total 600 timeslots. The demonstrated results are averaged over 20 instances with the 95% confidence level and standard deviation is included in all graphs.

### B. The Performance of Control Plane

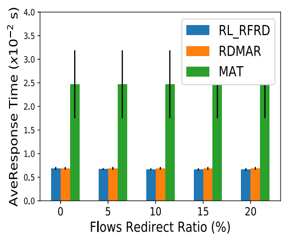
We mainly evaluate the effect of average response (AveResponse) time and controller fair index (CFI) [29] over the flows redirect ratio, which indicates the percentage of flows that has been selected to redirect. CFI refers to the load distribution among all controllers in the network and it is calculated by CFI $= \frac{(\sum_{c\in C}\theta_c)^2}{|C|\cdot\sum_{c\in C}{\theta_c}^2}$. Following [29], AveResponse time of controller $c$ can be derived by $RT_c = \frac{1}{\alpha_c - \theta_c}\mathcal{O}(n^2)$ where $n$ is the number of network switches. The value of $\mathcal{O}(n^2)$ can be measured through running a routing algorithm (e.g., OSPF) with $n$ nodes. The number of macroflows for each timeslot is set to 20.

Fig. 5 plots the load distribution among multiple controllers and it shows that both RL_RFRD and RDMAR can achieve good balance distribution. As shown in Fig. 5(a) and Fig. 5(b), CFI has improved 8.1% in ITALYNET and 4.2% in Fat-tree compared with RDMAR scheme after redirecting 20% flows. This is because in RDMAR, the mapping between switches and controllers is static and flows are waited to be handled

[1] https://github.com/iris0102/ModelTest

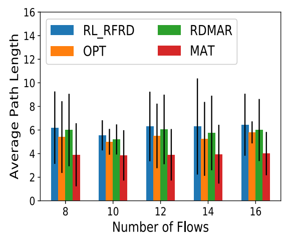


(a) ITALYNET

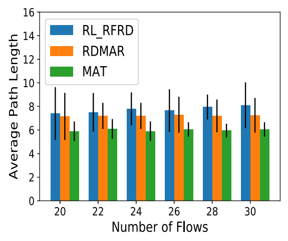


(b) Fat-tree

Fig. 6. Average response time vs. Flows redirect ratio.

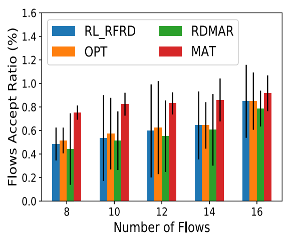


(a) ITALYNET

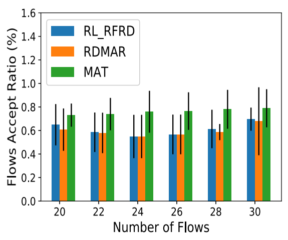


(b) Fat-tree

Fig. 8. Average path length vs. Number of flows.

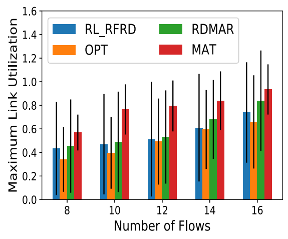


(a) ITALYNET

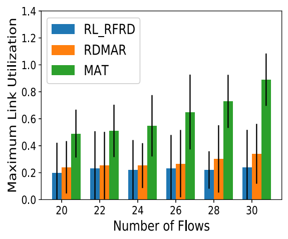


(b) Fat-tree

Fig. 7. Flows accept ratio vs. Number of flows.

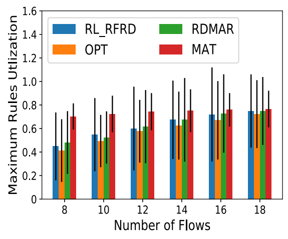


(a) ITALYNET

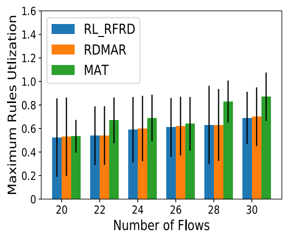


(b) Fat-tree

Fig. 9. Maximum link utilization vs. Number of flows.



(a) ITALYNET



(b) Fat-tree

Fig. 10. Maximum flow rules utilization vs. Number of flows.

when the current controller has no efficient capacity, but for RL_RFRD, some flows will be redirected to the controller with largest remaining processing capacity. Besides, as shown in Fig. 6(a) and Fig. 6(b), the average response time has improved 9.7% in ITALYNET and 14.1% in Fat-tree after redirecting 20% flows compared with RDMAR. In addition, Fig. 6 shows that MAT has more than 2x response time compared to RL_RFRD and RDMAR, because the controller capacity in MAT is bottleneck resource and it handles flows one by one.

## *C. The Performance of Data Plane*

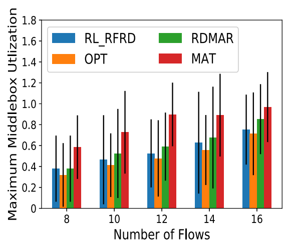
Since the Gurobi model is formulated to solve link load balancing, the optimum (OPT) value where the routing path is solved by Gurobi model is added in ITALYNET to evaluate the performance of data plane.

Fig. 7 depicts the ratio of number of flows requests accepted over the number of flows. Assume that the flows in the network can last for long time, it can be observed that MAT accepts more flows than that of multiple SDNs. This is because although MAT handles flow requests one by one, resource will be released after finishing routing so that MAT will have efficient network capacity to find the suitable routing path for each flow. In addition, RL_RFRD has higher flows accept ratio than RDMAR and it has closer performance with the OPT, when the number of flows is 14 in Fig. 7(a) and 24 in Fig. 7(b).

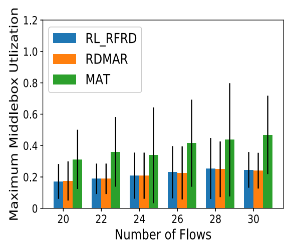
Fig. 8(a) and Fig. 8(b) plot the average routing path length over the number of flows in both topologies. Obviously, MAT has the shorter path length than that of multiple SDNs because it handles flow request one by one and always tries to find the short routing path for each flow under the network constraints. Compared with RDMAR, RL_RFRD will make flows to communicate in a longer way with light-loaded controllers. Thus, RL_RFRD generally has longer routing path length than RDMAR and MAT. Generally, when there is the switch attached middlebox in redirecting flow path satisfying the network functions for this flow, RL_RFRD may have the chance to outperform RDMAR in terms of path length, otherwise, it may take longer path to meet the routing policy.

Fig. 9 shows the maximum link utilization over the number of flows in both topologies. It can be observed from Fig. 9(a) and Fig. 9(b) that the link utilization will increase with the number of flows increasing as indicated by MAT, and more links will be occupied. With the advantage of balancing maximum link load, we observe that RL_RFRD has closer performance to OPT compared with RDMAR. This is because default path is properly assigned in advance when redirecting flows, so the maximum link load of RL_RFRD can be smaller than that of RDMAR.

It can be also observed from Fig. 10(a) and Fig. 10(b) that with the number of flows increasing, MAT requires more flow entries and it always keeps the higher maximum flow rules utilization compared with RL_RFRD and RDMAR. This is because MAT handles flow requests one by one and it takes more response time to assign routing path for flows, which may result in some flow entries expired and new flow rules to be installed. Similarly, Fig. 11(a) and Fig. 11(b) show that MAT has maximum middlebox utilization with the increasing number of flows.



(a) ITALYNET



(b) Fat-tree

Fig. 11. Maximum middlebox capacity utilization vs. Number of flows.

TABLE III
THE AVERAGE RUNNING TIME FOR EACH TIME SLOT (SECONDS)

| Topology | RL_RFRD | MAT | RDMAR | OPT |
|---|---|---|---|---|
| ITALYNET | 0.303 | 1.21 | 1.075 | 2.152 |
| Fat-tree | 0.418 | 7.381 | 0.499 | - |

### *D. The Performance of Running Time*

Table III shows the average running time for each time slot of three algorithms. Obviously, OPT has the highest running time compared with other three algorithms, because it searches the optimal solution in the global feasible set. Compared with MAT and RDMAR, RL_RFRD has lower running time since default paths for selected flows are deployed in both topologies. Compared with MAT, RL_RFRD and RDMAR are only responsible for its local controlled switches, which can reduce the range for routing decision. Besides, Fat-tree has higher time efficiency than that of ITALYNET where time efficiency is calculated over the running time of MAT, because Fat-tree can also reduce the input size for routing decision once the source and destination nodes are determined. This also indicates that RL_RFRD may also be applied in large-scale network.

## VI. CONCLUSION AND FUTURE WORK

For the reason that traffic dynamics will not not only pose a severe challenge to controller load balancing, but also affect the link load in the data plane, and network functions in middleboxes also may change the volume of processed traffic as well, this article focus on both controller load balancing and link load balancing with consolidated middleboxes. This problem is formulated as an integer programming optimization with the objective to minimize maximum controller load and link load, and the problem is proved to be NP-hard. In order to solve this problem, we have proposed a two-phase algorithm. The first phase is to redirect selected flows to controller with maximum processing capacity and the second phase is to find the fine-balanced middleboxes selection and routing path for each flow by rounding-based algorithm. Finally, numerical results show that our proposed RL_RFRD algorithm outperforms other algorithms in terms of load-balancing performance and response time. In future work, the path length can be included in the optimization objective because longer paths will consume large amount of network resources, so there is a tradeoff between load balancing and shortening path length.

## APPENDIX A
## PROOF OF LEMMA 1

We give two famous lemmas for probability analysis.

*Lemma 4 (Chernoff Bound):* Given $n$ independent variables: $x_1, x_2, \ldots, x_n$, where $\forall x_i \in [0,1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^{n} x_i]$. Then, $\mathbf{Pr}[\sum_{i=1}^{n} x_i \geq (1+\epsilon)\mu] \leq e^{\frac{-\epsilon^2\mu}{2+\epsilon}}$, where $\epsilon$ is an arbitrarily positive value.

*Lemma 5 (Union Bound):* Given a countable set of $n$ events: $A_1, A_2, \ldots, A_n$, each event $A_i$ happens with possibility $\mathbf{Pr}(A_i)$. Then, $\mathbf{Pr}(A_1 \cup A_2 \cup \cdots \cup A_n) \leq \sum_{i=1}^{n} \mathbf{Pr}(A_i)$.

*Proof:* The first step of L_RD algorithm will derive a fractional solution $\hat{y}^l_{(u,v)}$ for the relaxed s-JTMSR problem. Using the randomized rounding method, for each flow $f \in \mathcal{F}$, only one path in $\mathcal{P}_f$ will be chosen as its directing path. Thus, the traffic load of link $(u,v)$ from flow $f$ is defined as a random variable $\chi_{(u,v),f}$, which is expressed

$$\chi_{(u,v),f} = \begin{cases} d_f^{(u,v)}, & \text{with probability of } \sum_{(u,v)\in l: l\in\mathcal{P}_f} \hat{y}^l_{(u,v)} \\ 0, & \text{otherwise} \end{cases}$$

According to the definition, $\chi_{(u,v),f}$, with $f \in \mathcal{F}$ are mutually independent. Thus, the expected traffic load on link $(u,\ v)$ is:

$$\begin{aligned}\mathbb{E}\left[\sum_{f\in\mathcal{F}} \chi_{(u,v),f}\right] &= \sum_{f\in\mathcal{F}} \mathbb{E}\left[\chi_{(u,v),f}\right] \\ &= \sum_{f\in\mathcal{F}} \sum_{(u,v)\in l: l\in\mathcal{P}_f} \hat{y}^l_{(u,v)} \cdot d_f^{(u,v)} \leq B_{(u,v)}\end{aligned} \tag{27}$$

Combining Eq. (27) and the definition of $\alpha_0$ in Eq. (26), we have

$$\begin{cases} \frac{\chi_{(u,v),f}\cdot\alpha_0}{B_{(u,v)}} \in [0,1] \\ \mathbb{E}\left[\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}\cdot\alpha_0}{B_{(u,v)}}\right] \leq \alpha_0 \end{cases} \tag{28}$$

Then, by applying Lemma 4, assume that $\rho$ is an arbitrary positive value, it follows

$$\mathbf{Pr}\left[\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}\cdot\alpha_0}{B_{(u,v)}} \geq (1+\rho)\cdot\alpha_0\right] \leq e^{\frac{-\rho^2\alpha_0}{2+\rho}}$$

Now we assume that

$$\mathbf{Pr}\left[\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}}{B_{(u,v)}} \geq (1+\rho)\right] \leq e^{\frac{-\rho^2\alpha_0}{2+\rho}} \leq \frac{\mathcal{H}}{|\mathrm{V}|^2} \tag{29}$$

where $\mathcal{H}$ is the function of network-related variables such as the number of switches $|\mathrm{V}|$, and $\mathcal{H} \to 0$ when the size of network grows.

The solution for Eq. (29) is expressed as:

$$\rho \geq \frac{\log\frac{|\mathrm{V}|^2}{\mathcal{H}} + \sqrt{\log^2\frac{|\mathrm{V}|^2}{\mathcal{H}} + 8\cdot\alpha_0\log\frac{|\mathrm{V}|^2}{\mathcal{H}}}}{2\cdot\alpha_0},\ |\mathrm{V}| \geq 2 \tag{30}$$

Then, set $\mathcal{H} = \frac{1}{|\mathrm{V}|^2}$. Eq. (29) is transformed into:

$$\mathbf{Pr}\left[\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}}{B_{(u,v)}} \geq (1+\rho)\right] \leq e^{\frac{-\rho^2\alpha_0}{2+\rho}} \leq \frac{1}{|\mathrm{V}|^4}, \quad \text{where } \rho \geq \frac{4\log|\mathrm{V}|}{\alpha_0}+2$$

By applying Lemma 5, we have

$$\begin{aligned}\mathbf{Pr}&\left[\bigvee_{(u,v)\in\mathrm{E}}\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}}{B_{(u,v)}} \geq (1+\rho)\right]\\ &\leq \sum_{(u,v)\in\mathrm{E}}\mathbf{Pr}\left[\sum_{f\in\mathcal{F}} \frac{\chi_{(u,v),f}}{B_{(u,v)}} \geq (1+\rho)\right]\\ &\leq |\mathrm{V}|^2\cdot\frac{1}{|\mathrm{V}|^4} = \frac{1}{|\mathrm{V}|^2},\ \rho \geq \frac{4\log|\mathrm{V}|}{\alpha_0}+2 \qquad (31)\end{aligned}$$

Note that the second inequality in Eq. (31) holds since there are at most $|\mathrm{V}|^2$ links in the network. Thus, Eq. (29) is guaranteed with $1+\rho = \frac{4\log|\mathrm{V}|}{\alpha_0}+3$, which concludes the proof. ■

## Appendix B
## Proof of Lemma 2

*Proof:* We define a random variable $z_{u,f}$ to formulate the required flow entry on a switch $u \in \mathrm{V}$ for directing flow $f \in \mathcal{F}$,

$$z_{u,f} = \begin{cases} \tau_f, & \text{with probability of } \sum_{u\in l:l\in\mathcal{P}_f}\hat{y}^l_{(u,v)} \\ 0, & \text{otherwise}\end{cases}$$

By the definition, $z_{u,f}$, with $f \in \mathcal{F}$, are also independent. Thus, we have

$$\begin{cases} \frac{z_{u,f}\cdot\alpha_1}{T_u} \in [0,1] \\ \mathbb{E}\left[\sum_{f\in\mathcal{F}}\frac{z_{u,f}\cdot\alpha_1}{T_u}\right] \leq \alpha_1 \end{cases} \qquad (32)$$

Similar to the proof for the link capacity constraint, we have a general form of proving: the number of required flow entries on any switch *u* will not exceed the flow-table size constraint $T_u$ by a factor of $1+\varphi$ if and only if there exists $\mathcal{H}$ satisfying

$$\mathbf{Pr}\left[\bigvee_{u\in\mathrm{V}}\sum_{f\in\mathcal{F}}\frac{z_{u,f}\cdot\alpha_1}{T_u} \geq (1+\varphi)\cdot\alpha_1\right] \leq \mathcal{H} \qquad (33)$$

where $\varphi$ is the function of network-related variables such as the number of switches $|\mathrm{V}|$, and $\mathcal{H}$ is defined the same as that in Eq. (29). By applying Lemma 5, Eq. (33) is relaxed to:

$$\begin{aligned}\mathbf{Pr}&\left[\bigvee_{u\in\mathrm{V}}\sum_{f\in\mathcal{F}}\frac{z_{u,f}\cdot\alpha_1}{T_u} \geq (1+\varphi)\cdot\alpha_1\right]\\ &\leq \sum_{u\in\mathrm{V}}\mathbf{Pr}\left[\sum_{f\in\mathcal{F}}\frac{z_{u,f}\cdot\alpha_1}{T_u} \geq (1+\varphi)\cdot\alpha_1\right] \leq \mathcal{H}\end{aligned}$$

By applying Lemma 4 and Eq. (32), we can assume:

$$\begin{aligned}\mathbf{Pr}\left[\sum_{f\in\mathcal{F}}\frac{z_{u,f}\cdot\alpha_1}{T_u} \geq (1+\varphi)\cdot\alpha_1\right] &\leq e^{\frac{-\varphi^2\alpha_1}{2+\varphi}}\\ &\leq \frac{\mathcal{H}}{|\mathrm{V}|},\ |\mathrm{V}|\geq 2\end{aligned}$$

Then, we get the result:

$$\varphi \geq \frac{\log\frac{|\mathrm{V}|}{\mathcal{H}} + \sqrt{\log^2\frac{|\mathrm{V}|}{\mathcal{H}} + 8\cdot\alpha_1\log\frac{|\mathrm{V}|}{\mathcal{H}}}}{2\cdot\alpha_1},\ |\mathrm{V}|\geq 2 \qquad (34)$$

Set $\mathcal{H} = \frac{1}{|\mathrm{V}|^2}$. Apparently $\mathcal{H} \to 0$ as $|\mathrm{V}| \to \infty$. With respect to Eq. (34), we set

$$\begin{aligned}\varphi &= \frac{\log\frac{|\mathrm{V}|}{\mathcal{H}} + \log\frac{|\mathrm{V}|}{\mathcal{H}} + 4\cdot\alpha_1}{2\cdot\alpha_1}\\ &= \frac{6\log|\mathrm{V}|+4\cdot\alpha_1}{2\cdot\alpha_1} = \frac{3\log|\mathrm{V}|}{\alpha_1}+2 \qquad (35)\end{aligned}$$

Thus, Eq. (33) is guaranteed with a factor of $1+\varphi = \frac{3\log|\mathrm{V}|}{\alpha_1}+3$ and $\mathcal{H} = \frac{1}{|\mathrm{V}|^2}$, which concludes the proof. ■

## Appendix C
## Proof of Lemma 3

*Proof:* We define a random variable $\omega_{u,f}$ to indicate the resource consumed on a switch $u \in V_p$ for directing flow $f \in \mathcal{F}$,

$$\omega_{u,f} = \begin{cases} P^u_f, & \text{with probability of } \sum_{u\in l:l\in\mathcal{P}_f}\hat{y}^l_{(u,v)} \\ 0, & \text{otherwise}\end{cases}$$

By the definition, $\omega_{u,f}$, with $f \in \mathcal{F}$, are also independent. Thus, we have

$$\begin{cases} \frac{\omega_{u,f}\cdot\alpha_2}{P_u} \in [0,1] \\ \mathbb{E}\left[\sum_{f\in\mathcal{F}}\frac{\omega_{u,f}\cdot\alpha_2}{P_u}\right] \leq \alpha_2 \end{cases} \qquad (36)$$

Similar to the proof for the flow-table size capacity, we have a general form of proving: the resource consumed on any switch $u \in V_p$ will not exceed the middlebox capacity constraint $P_u$ by a factor of $1+\sigma$ if and only if there exists $\mathcal{H}$ satisfying

$$\mathbf{Pr}\left[\bigvee_{u\in V_p}\sum_{f\in\mathcal{F}}\frac{\omega_{u,f}\cdot\alpha_2}{P_u} \geq (1+\sigma)\cdot\alpha_2\right] \leq \mathcal{H} \qquad (37)$$

where $\sigma$ is the function of network-related variables such as the number of switches $|V_p|$ that directly connected to PMs, and $\mathcal{H}$ is defined the same as that in Eq. (29). By applying Lemma 5, Eq. (37) is relaxed to:

$$\begin{aligned}\mathbf{Pr}&\left[\bigvee_{u\in V_p}\sum_{f\in\mathcal{F}}\frac{\omega_{u,f}\cdot\alpha_2}{P_u} \geq (1+\sigma)\cdot\alpha_2\right]\\ &\leq \sum_{u\in V_p}\mathbf{Pr}\left[\sum_{f\in\mathcal{F}}\frac{\omega_{u,f}\cdot\alpha_2}{P_u} \geq (1+\sigma)\cdot\alpha_2\right] \leq \mathcal{H}\end{aligned}$$

By applying Lemma 4 and Eq. (36), we can assume:

$$\mathbf{Pr}\left[\sum_{f\in\mathcal{F}}\frac{\omega_{u,f}\cdot\alpha_2}{P_u}\geq(1+\sigma)\cdot\alpha_2\right]\leq e^{\frac{-\varphi^2\alpha_2}{2+\varphi}}$$
$$\leq\frac{\mathcal{H}}{|V_p|},\ |V_p|\geq 2$$

Then, we get the result:

$$\varphi\geq\frac{\log\frac{|V_p|}{\mathcal{H}}+\sqrt{\log^2\frac{|V_p|}{\mathcal{H}}+8\cdot\alpha_2\log\frac{|V_p|}{\mathcal{H}}}}{2\cdot\alpha_2},\ |V_p|\geq 2 \tag{38}$$

Set $\mathcal{H}=\frac{1}{|V_p|^2}$. Apparently $\mathcal{H}\to 0$ as $|V_p|\to\infty$. With respect to Eq. (38), we set

$$\varphi=\frac{\log\frac{|V_p|}{\mathcal{H}}+\log\frac{|V_p|}{\mathcal{H}}+4\cdot\alpha_2}{2\cdot\alpha_2}$$
$$=\frac{6\log|V_p|+4\cdot\alpha_2}{2\cdot\alpha_2}=\frac{3\log|V_p|}{\alpha_2}+2 \tag{39}$$

Thus, Eq. (37) is guaranteed with a factor of $1+\sigma=\frac{3\log|V_p|}{\alpha_2}+3$ and $\mathcal{H}=\frac{1}{|V_p|^2}$, which concludes the proof. ■

## REFERENCES

[1] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM 2012 Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2012, pp. 13–24.

[3] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. 9th USENIX NSDI Conf. Netw. Syst. Design Implement.*, 2012, pp. 323–336.

[4] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," *Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, 2013.

[5] Y. Chang, G. Petri, S. Rao, and T. Rompf, "Composing middlebox and traffic engineering policies in SDNs," in *Proc. IEEE INFOCOM Conf. Commun. Workshops*, Atlanta, GA, USA, 2017, pp. 1–9.

[6] B. Anwer, T. Benson, N. Feamster, and D. Levin, "Programming slick network functions," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, pp. 1–13.

[7] X. Li and C. Qian, "A survey of network function placement," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2016, pp. 948–953.

[8] M. Alqarni, A. Ing, and B. Tang, "LB-MAP: Load-balanced middlebox assignment in policy-driven data centers," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Vancouver, BC, Canada, 2017, pp. 1–9.

[9] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Funct. Virtualization (HotMiddlebox)*, 2015, pp. 55–60.

[10] M. Huang, W. Liang, Z. Xu, M. Jia, and S. Guo, "Throughput maximization in software-defined networks with consolidated middleboxes," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Dubai, UAE, 2016, pp. 298–306.

[11] H. Huang, S. Guo, J. Wu, and J. Li, "Joint middlebox selection and routing for software-defined networking," in *Proc. IEEE ICC Int. Conf. Commun.*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[12] P. Duan, Q. Li, Y. Jiang, and S. T. Xia, "Toward latency-aware dynamic middlebox scheduling," in *Proc. 24th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Las Vegas, NV, USA, 2015, pp. 1–8.

[13] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

[14] W. Ma *et al.*, "SDN-based traffic aware placement of NFV middleboxes," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 528–542, Sep. 2017.

[15] W. Ma, J. Beltran, D. Pan, and N. Pissinou, "Placing traffic-changing and partially-ordered NFV middleboxes via SDN," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1303–1317, Dec. 2019.

[16] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 560–573, Jul./Aug. 2017.

[17] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *Proc. 21st IEEE Int. Workshop Local Metropolitan Area Netw.*, Beijing, China, 2015, pp. 1–6.

[18] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. Conf. Comput. Commun.*, Kowloon, Hong Kong, 2015, pp. 1346–1354.

[19] B. Kar, E. H.-K. Wu, and Y.-D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 372–386, Mar. 2018.

[20] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, 2016, pp. 1–9.

[21] M. C. Luizelli, D. Raz, and Y. Sa'ar, "Optimizing NFV chain deployment through minimizing the cost of virtual switching," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 2150–2158.

[22] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNext)*, 2014, pp. 271–282.

[23] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "FlowTags: Enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 19–24.

[24] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers," in *Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun.*, San Francisco, CA, USA, 2016, pp. 1–9.

[25] G. Cheng, H. Chen, Z. Wang, and S. Chen, "DHA: Distributed decisions on the switch migration toward a scalable SDN control plane," in *Proc. IFIP Netw. Conf.*, Toulouse, France, 2015, pp. 1–9.

[26] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, "A switch migration-based decision-making scheme for balancing load in SDN," *IEEE Access*, vol. 5, pp. 4537–4544, 2017.

[27] Y. Xu *et al.*, "Dynamic switch migration in distributed software-defined networks to achieve controller load balance," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 515–529, Mar. 2019.

[28] H. Wang, H. Xu, L. Huang, J. Wang, and X. Yang, "Load-balancing routing in software defined networks with multiple controllers," *Comput. Netw.*, vol. 141, pp. 82–91, Aug. 2018.

[29] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, and Y. Sun, "Minimizing controller response time through flow redirecting in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 562–575, Feb. 2018.

[30] G. Sallam and B. Ji, "Joint placement and allocation of virtual network functions with budget and capacity constraints," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, 2019, pp. 523–531.

[31] O. Alhussein *et al.*, "A virtual network customization framework for multicast services in NFV-enabled core networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1025–1039, Jun. 2020.

[32] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[33] J. Shi, J. Wang, H. Huang, L. Shen, J. Zhang, and H. Xu, "Joint optimization of stateful VNF placement and routing scheduling in software-defined networks," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl. Ubiquitous Comput. Commun. Big Data Cloud Comput. Soc. Comput. Netw. Sustain. Comput. Commun. (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, Melbourne, VIC, Australia, 2019, pp. 9–14.

[34] M. C. Luizelli, D. Raz, Y. Sa'ar, and J. Yallouz, "The actual cost of software switching for NFV chaining," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manag. (IM)*, Lisbon, Portugal, 2017, pp. 335–343.

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

[36] H. Li, R. E. De Grande, and A. Boukerche, "An efficient CPP solution for resilience-oriented SDN controller deployment," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Lake Buena Vista, FL, USA, 2017, pp. 540–549.

[37] M. He, A. Varasteh, and W. Kellerer, "Toward a flexible design of SDN dynamic control plane: An online optimization approach," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1694–1708, Dec. 2019.

[38] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2013, pp. 121–126.

[39] L. Dan, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: State distribution trade-offs in software defined networks," in *Proc. 1st ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 1–6.

[40] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 459–473.

[41] C. H. Chi, J. Deng, and Y. H. Lim, "Compression proxy server: Design and implementation," in *Proc. 2nd Conf. USENIX Symp. Internet Technol. Syst. (USITS)*, 1999, pp. 1–12.

[42] X. Chen, M. Sterna, X. Han, and J. Blazewicz, "Scheduling on parallel identical machines with late work criterion: Offline and online cases," *J. Sched.*, vol. 19, no. 6, pp. 729–736, 2016.

[43] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide To The Theory of NP-Completeness*. San Francisco, CA, USA: W. H. Freeman, 1979.

[44] G. Zhao, H. Xu, S. Chen, L. Huang, and P. Wang, "Joint optimization of flow table and group table for default paths in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1837–1850, Aug. 2018.

[45] P. Li, S. Guo, C. Pan, L. Yang, G. Liu, and Y. Zeng, "Fast congestion-free consistent flow forwarding rules update in software defined networking," *Future Gener. Comput. Syst.*, vol. 97, pp. 743–754, Aug. 2019.

[46] A. Capone, C. Cascone, A. Q. T. Nguyen, and B. Sansò, "Detour planning for fast and reliable failure recovery in SDN with openstate," in *Proc. 11th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, Kansas City, MO, USA, 2015, pp. 25–32.

[47] M. Chen, X. Yu, and Y. Liu, "PCNN: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3550–3559, Nov. 2018.

[48] V. Dukic, S. A. Jyothi, B. Karlas, M. Owaida, C. Zhang, and A. Singla, "Is advance knowledge of flow sizes a plausible assumption?" in *Proc. 16th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 565–580.

[49] Y. Yao, S. Guo, P. Li, G. Liu, and Y. Zeng, "Forecasting assisted VNF scaling in NFV-enabled networks," *Comput. Netw.*, vol. 168, pp. 1–13, Feb. 2020.

[50] H. Huang, D. Zeng, S. Guo, and H. Yao, "Joint optimization of task mapping and routing for service provisioning in distributed datacenters," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, 2014, pp. 4196–4201.

[51] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 63–74.

[52] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2010, pp. 267–280.

[53] *Gurobi Optimizer Reference Manual*, Gurobi Optim., Beaverton, OR, USA, 2015.