IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 22, 2020, accepted July 29, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013770

End-to-End Delay Minimization-Based Joint Rule Caching and Flow Forwarding Algorithm for SDN

LEI LUO[1], RONG CHAI[1], (Senior Member, IEEE), QIONGFANG YUAN[1], JINYAN LI[2], AND CHENGLI MEI[2]

[1]School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2]Technology Innovation Center, China Telecom Corporation Ltd., Beijing 102209, China

Corresponding author: Rong Chai (chairong@cqupt.edu.cn)

This work was supported by the National Science and Technology Specific Project of China under Grant 2018ZX03001016.

ABSTRACT Software-defined networking (SDN) technology is expected to offer higher flexibility and programmability and enhanced transmission performance by decoupling control plane from data plane and enabling centralized network management. In SDN, switches may cache a certain number of flow forwarding rules, so that user flows can be forwarded accordingly. In this article, stressing the limited caching space of switches and the heterogeneous transmission performance of switches and links, we jointly design rule caching and flow forwarding strategy for multiple user flows in SDN. To emphasize the importance of the end-to-end delay caused by the transmission and processing of user flows in both the data plane and control plane, we formulate the joint optimization problem as an end-to-end delay minimization problem. As the original optimization problem is a non-deterministic polynomial hard (NP-hard) problem, which cannot be solved directly, we propose a heuristic algorithm which successively solves three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem, and resource sharing subproblem. By applying the K-shortest path algorithm, a priority-based rule caching algorithm, and Lagrangian dual method, respectively, the three subproblems are solved and the joint rule caching and flow forwarding strategy is obtained. Simulation experiments are conducted to examine the effectiveness of the proposed algorithm, and the results indicate that our proposed algorithm is capable of improving system performance by about 20% compared with the previous solutions.

INDEX TERMS Software-defined networking, end-to-end delay, rule caching, flow forwarding, resource sharing.

I. INTRODUCTION

The vertically integrated network architecture of traditional Internet and distributed packet forwarding schemes result in complicated network control and (re)configuration, and highly limited packet transmission performance [1]. To overcome the limitations of traditional network architectures, a brand-new networking paradigm named software-defined networking (SDN) was proposed, which decouples control plane from data plane and enables flexible configuration and management of network resources [2], [3]. More specifically, the control layer of SDN containing one or multiple controllers is responsible for monitoring network states and determining network management strategies, such as the forward strategies of user flows, and the data plane of SDN is mainly composed of switches and routers, and is responsible for performing user data forwarding according to the forwarding strategies disseminated by controllers [4]. Benefited from the simplified design of data forwarding devices and the direct and intentional control over network behavior at controllers, SDN is capable of facilitating flexible network management, and offering users services with desired quality of service (QoS) [5].

In SDN, controllers may determine user flow forwarding strategies and cache certain flow forwarding rules at the ternary content addressable memory (TCAM) of switches [6]. When user flows arrive at SDN switches, the switches match their pre-captured forwarding rules with the received user flows. If the corresponding forwarding rules can be found, the switches will forward user flows accordingly.

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

While caching flow forwarding rules at TCAM excels in packet processing, due to the diverse flow transmission requirements and limited cache spaces of switches, it is highly impossible to cache the rules of all user flows at switches. In the case that no forwarding rules of one user flow can be found in their TCAM, the switches having received the user flow will send flow forwarding request messages, referred to as packet-in messages, to its associated controller, which calculates and determines flow forwarding strategies and disseminates packet-out messages containing the obtained strategy to the switches along the transmission path of the user flow [7]. In order to achieve efficient utilization of TCAM as well as enhanced transmission performance of user flows, reasonable and effective rule caching schemes should be designed for SDN.

In an attempt to simplify forwarding devices and facilitate centralized network management, the centralized controllers in SDN are in charge of designing flow forwarding strategies for user flows. On account of the available global network view information, the flow forwarding strategies designed by centralized controllers are expected to be more efficient and far-sighted compared to those made in distributed networks. However, due to the heterogeneous characteristics of switches and transmission links in SDN, and the diverse requirements of user flows, designing flow forwarding schemes which achieve the performance optimization of multiple user flows is challenging [8].

Rule caching and flow forwarding schemes have been studied for SDN in previous work independently [9]–[27], however, the joint design of rule caching and flow forwarding schemes has not been studied extensively. In particular, to design flow forwarding strategy, previous work mainly ignores the issue of rule caching, assumes that no flow forwarding rules have been cached and designs flow forwarding strategy for user flows. On the other hand, to design rule caching strategy, previous work either assumes that flow forwarding strategy is given or selects the switches with relatively high performance to cache rules and skipped flow forwarding issues.

As a matter of fact, the two issues, i.e., rule caching and flow forwarding are indeed closely coupled. For instance, in the case that the forwarding rules of a specific user flow have been cached at one switch, selecting the switch to forward the user flow would be preferred. As the switch is capable of forwarding the flow directly according to the cached rules, instead of querying SDN controllers for forwarding strategy, the flow forwarding performance can be enhanced especially in terms of processing delay. Similarly, if one switch is selected as the forwarding switch of one user flow, caching the flow forwarding rules of the user flow at the switch would be highly desired in order to improve the efficiency of rule caching and the utilization of caching space. Therefore, taking into account the correlation between rule caching and flow forwarding, and designing joint strategy is of particular importance.

In this article, the problem of joint rule caching and flow forwarding for multiple user flows in SDN is investigated. To emphasize the importance of the end-to-end delay caused by the transmission and processing delay of user flows at switches and controllers, we formulate the joint optimization problem as an end-to-end delay minimization problem. As the original optimization problem is non-deterministic polynomial hard (NP-hard), which cannot be solved directly, we transform the formulated problem into three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem, and resource sharing subproblem. Applying the K-shortest path algorithm, priority-based rule caching algorithm and the Lagrangian dual method to solve the three subproblems respectively, we obtain the joint rule caching and flow forwarding strategy.

The major contributions of this article are summarized as follows.

1) In this article, we consider the data transmission problem of a number of user flows in SDN. To achieve highly utilized rule caching at the TCAM of switches and efficient forwarding of user flows, we investigate the joint problem of rule caching and flow forwarding.
2) While rule caching and flow forwarding schemes have been studied for SDN in previous work independently [9]–[27], it can be demonstrated that the two issues are closely related and the joint design is highly desired. Although the authors in [28]–[31] stressed both the rule caching and flow forwarding issues, they mainly assumed that flow forwarding strategy was given, and designed the optimal rule caching strategy in order to minimize caching space occupation [28], [29] or maximize the utilization of caching space [30], [31], and failed to consider extensively the overall network performance, especially the end-to-end delay of user flows. Unlike previous work, to emphasize the importance of the end-to-end delay caused by the transmission and processing delay of user flows in both data plane and control plane, we take into account flow forwarding requirements, rule caching and resource allocation constraints, and formulate the joint rule caching and flow forwarding problem as an end-to-end delay minimization problem.
3) As the original optimization problem is an NP-hard problem, which cannot be solved directly, we transform the problem into three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem and resource sharing subproblem. By applying the K-shortest path algorithm, priority-based rule caching algorithm and Lagrangian dual method, respectively, the three subproblems are solved and the joint rule caching and flow forwarding strategy is obtained. Simulation experiments show that our proposed algorithm is capable of improving system performance by about 20% compared with the previous solutions.

The structure of this article is as follows. In Section II, we review the related work. In Section III, we present

system model. Section IV formulates the optimization problem. In Section V, we discuss the solution to the optimization problem. Complexity analysis of the proposed algorithm is presented in Section VI. Section VII illustrates the simulation results. In Section VIII, we discuss the innovations and limitations of the proposed algorithm. Finally, we make a conclusion in Section IX.

II. RELATED WORK

In this section, we present an overview of related work in recent literature.

A. RULE CACHING ALGORITHMS FOR SDN

Most of the previously proposed rule caching algorithms attempt to optimize network cost or cache hit rate [9]–[16]. To jointly consider the information processing delay of control plane and the utilization of TCAM, the authors in [9] formulated a cost optimization problem and obtained a rule caching scheme by solving the problem. Reference [10] addresses the TCAM cache replacement problem in SDN. Jointly considering cache hit rate and the cost required for caching rules, the authors proposed an effective cache replacement algorithm, which tends to replace the rules with fewer hits while taking up large space. In order to reduce rule update cost, the authors in [11] proposed a traffic-aware hybrid rule allocation scheme which dynamically executes reactive caching and proactive caching according to network traffic states.

In an attempt to improve the utilization rate of TCAM, the authors in [12] proposed a timeout scheme which allocates various timeout values for forwarding rules based on data plane features and flow dynamics, and presented a hybrid Q-learning algorithm to determine the rule caching strategy. Reference [13] studies the problem of wildcard rule caching and cache replacement in SDN. Initially, the frequently matched wildcard rules with relatively less extra cache cost were cached, then, once cache miss occurred, the cached rules would be replaced by the newly missed rules so as to improve cache hit rate. In [14], the authors predicted replacement rules based on the arrival interval distribution model of user flows, and proposed a TCAM management scheme to reduce link transmission delay between switches and controllers.

The problem of flow table space allocation was investigated in [15]. Jointly considering the caching cost of flow table and the fairness among user flows, the authors formulated the flow table space allocation problem as a cache cost minimization problem under the minimum-maximum fairness constraint. A two-stage heuristic algorithm was proposed to solve the optimization problem and obtain the flow table space allocation strategy. To reduce the processing delay at control plane of SDN and improve the utilization of TCAM space in the meantime, [16] proposes a rule partition and distribution algorithm which divides the set of flow rules into predetermined number of disjoint subsets and distributes the subsets uniformly to the switches.

In [17], it is assumed that due to the asynchronous nature of SDN, packets may reach an intermediate switch before the corresponding flow rules. In this case, the packet is dropped by the switch. To address this problem, the authors computed the delay required for installing flow rules and transmitting packets along the path. In the case that the packet arrival delay at one switch is less than the delay of corresponding flow rule installation at the switch, the packet is delayed for a minimum duration at the predecessor switch of the switch so as to guarantee that the corresponding flow rule has been installed before packet arrival.

B. FLOW FORWARDING ALGORITHMS OF SDN

In recent years, various flow forwarding strategies have been designed for SDN [18]–[27]. Transmission delay was addressed in designing flow forwarding strategies for SDN [18]–[20]. Considering the strict delay requirement of video flows, [18] proposes a delay-optimized flow forwarding algorithm that clearly distinguishes the forwarding strategies of different user flows based on their delay sensitivity. In [19], to deal with the forwarding path congestion problem of multiple user flows, the authors proposed a greedy algorithm according to the resource dependency graph and presented a time-extended network construction to reduce the update delay in the network. Reference [20] stresses flow forwarding and update problem in SDN. The flow forwarding problem was formulated as a delay minimization problem and solved by using randomized rounding method. A number of user flows with higher traffic volume were chosen for route update under link capacity constraint.

Tackling the rapidly increasing energy consumption in networks, energy consumption-aware flow forwarding algorithms were proposed [21]–[23]. The authors in [21], [22] proposed to save the energy consumption of networks by switching off a number of data forwarding devices. Reference [21] dynamically designs the flow forwarding path according to traffic demands, and achieved the energy saving in data center networks (DCNs) by closing inactive network equipments. In [22], the problem of energy-aware flow forwarding routing in SDN-based Ethernet networks was addressed. Under the constraint of rule space capacity of switches, the authors proposed network nodes and links turning off strategies so as to reduce energy consumption. In order to save the energy consumption of networks, [23] formulated an energy consumption minimization problem and solved it through applying an N algorithm-based joint routing and flow allocation scheme.

To achieve load balancing among switches, various flow forwarding strategies have been designed [24]–[26]. The authors in [24] proposed a vswitch deployment scheme for SDN, and formulated the joint flow forwarding and vswitch deployment problem as a load balancing problem, which was solved by a rounding-based scheme with bounded approximation factors. Flow forwarding problem in an SDN-based fat-tree DCN was studied in [25]. A low-cost load balancing flow forwarding framework was proposed

which dynamically computes load-deviation parameter of switches, and adjusts flow forwarding strategy accordingly to achieve load balancing among switches. In [26], the authors formulated the multi-path flow scheduling problem in SDN as a multi-objective optimization problem. A heuristic traffic balancing algorithm was proposed that periodically monitors network links and dynamically switches the flows on heavy links to the switches with relatively light loads.

In [27], the authors proposed a distributed flow architecture for networked enterprises. In the architecture, the controller distributes the rules across (a subset of) the switches, called ''authority switches,'' so as to scale the network to large topologies with many rules. The controller runs a partitioning algorithm that divides the rules evenly and minimizes fragmentation of the rules across multiple authority switches. The switches handle all packets in the data plane and divert packets through authority switches as needed to access the appropriate rules.

C. JOINT RULE CACHING AND FLOW FORWARDING ALGORITHMS OF SDN

In previous research work, the problems of rule caching and flow forwarding have been studied independently [9]–[27]. However, these two issues are indeed closely coupled and it can be demonstrated that by jointly designing the two issues, the performance of rule caching and flow forwarding can be enhanced significantly [28]–[31].

By allowing rule multiplexing in TCAM, a joint flow forwarding and rule caching algorithm was proposed in [28]. The authors first assumed that a list of candidate paths is given, and designed rule caching strategy so as to minimize the space occupation of the TCAM. They then extended the problem and formulated the joint flow routing and rule placement problem. A heuristic algorithm using relaxation and rounding techniques was presented, and feasible solutions were obtained by invoking the proposed PathSearch, PathRulePlacement, and SessionRulePlacement algorithms, respectively. Under the constraint of the maximum transmission delay in the network, [29] jointly considers the problem of flow forwarding and rule update, and introduces a garbage collection technique to minimize the space occupation of the switches.

The authors in [30] proposed an adaptive flow rule caching strategy in SDN. The proposed scheme consists of three phases, i.e., forwarding path selection, flow-rule placement, and rule redistribution. In the first phase, the authors formulated a max-flow-min-cost optimization problem to determine optimal forwarding paths. In the second phase, an integer linear programming problem was formulated to decide forwarding rules for candidate paths, so that the total number of matching rules was minimized. Finally, a rule redistribution scheme was proposed to improve the utilization of rule caching. The authors in [31] proposed an SDN-based framework which supports multi-flow transport protocols. A joint multi-flow forwarding and rule caching problem was formulated and solved to satisfy delay and throughput requirements of users.

While the authors in [28]–[31] considered rule caching and flow forwarding issues, they mainly assumed that flow forwarding strategy was given, and designed the optimal rule caching strategy in order to minimize caching space occupation [28], [29] or maximize the utilization of caching space [30], [31]. In this article, we investigate the problem of joint rule caching and flow forwarding for multiple user flows in SDN, and formulate the joint optimization problem as an end-to-end delay minimization problem. By solving the formulated optimization problem, the joint rule caching and flow forwarding strategy can be obtained.

III. SYSTEM MODEL

A. NETWORK SCENARIO

This work considers an SDN network composed of an SDN controller and multiple SDN switches. Let N be the number of switches, and V_i denote the ith SDN switch, $1 \leq i \leq N$. We model the SDN network as an undirected graph $G = \{V, E\}$, where $V = \{V_i\}$ is the set of nodes, $E = \{E_{i,j}\}$ is the set of edges, $E_{i,j}$ is the link connecting V_i and V_j, $1 \leq i, j \leq N,\ i \neq j$.

Let C_i be the TCAM storage size of V_i, $1 \leq i \leq N$. Assuming that each link has a maximum transmission rate, also referred to as link capacity, which is determined by the port rate of V_i and V_j, we denote the capacity of $E_{i,j}$ as $B_{i,j}$, $1 \leq i, j \leq N,\ i \neq j$. In order to characterize the state of physical connection between switches, we introduce $x_{i,j} \in \{0, 1\}$ to represent the connection identification variable between V_i and V_j. If V_i is a neighboring node of V_j, then $x_{i,j} = 1$, otherwise, $x_{i,j} = 0$, $1 \leq i, j \leq N$, $i \neq j$. In this work, the network topology is known, i.e., $x_{i,j}$ is a given constant.

B. CHARACTERISTICS OF USER FLOW TRANSMISSION

Suppose user flows with fixed traffic demand are required to be transmitted from their predefined source switches to destination switches through the network. Assuming that there are L user flows in the SDN network, we denote f_l as the lth user flow, $1 \leq l \leq L$. For convenience, the source switch and the destination switch of f_l are denoted by S_l and T_l, respectively. We denote Q_l as the traffic demand of f_l and R_l as the number of forwarding rules of f_l, $1 \leq l \leq L$. Considering the QoS requirement of user flows on transmission data rate, we assume that user flows may have various minimum rate requirement and denote $B_l^{\min}$ as the minimum rate requirement of f_l, $1 \leq l \leq L$.

C. AN EXAMPLE OF FLOW TRANSMISSION

Figure 1 shows the system model. As an example, in the figure, we assume that user flow f_2 with specific traffic demand needs to be transmitted from S_2 to T_2 and a multi-hop path from S_2 to T_2 via V_2 and V_3 has been selected for f_2. Upon receiving f_2, V_2 checks its TCAM. In the case that the flow forwarding rules of f_2 have been pre-cached at the

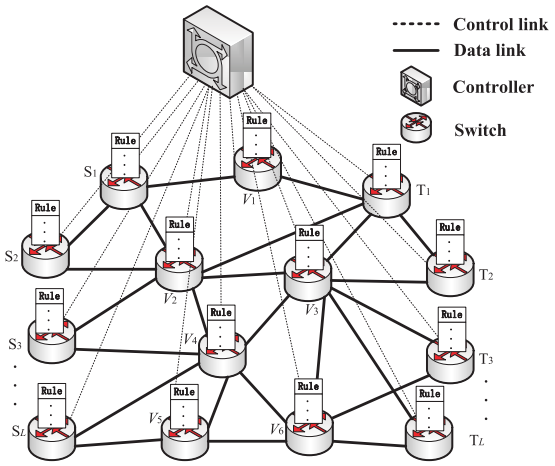

FIGURE 1. System model.

TCAM of V_2, indicating that f_2 should be forwarded to V_3, V_2 will forward f_2 directly to V_3. On the contrary, if no flow forwarding rule of f_2 has been pre-cached at the TCAM of V_2, V_2 should send a packet-in message to the controller, which determines the flow forwarding strategy by conducting the proposed algorithm. Then the controller will disseminate the obtained strategy to the switches along the forwarding path of f_2, i.e., V_2 and V_3. Based on the received strategy, V_2 and V_3 perform flow forwarding for f_2 accordingly.

In this article, we assume that various flow forwarding rules can be stored at the TCAM of switches, and jointly design rule caching and flow forwarding strategy for user flows. To facilitate efficient transmission of user flows, we assume that network resources, including the capacity resource of links and storage resource of switches are allowed to be shared among multiple user flows.

IV. END-TO-END DELAY OPTIMIZATION PROBLEM FORMULATION

In this article, aiming to optimize the total end-to-end delay of user flows, we model the joint rule caching and flow forwarding problem as a delay minimization problem subject to a number of optimization constraints. The detailed optimization problem formulation is shown below.

A. END-TO-END DELAY FORMULATION

Considering the overall network performance, we let D denote the overall delay of all user flows in the network, and express D as

$$D = \sum_{l=1}^{L} D_l \tag{1}$$

where D_l represents the total end-to-end delay required to transmit f_l from S_l to T_l. D_l consists of data plane delay and control plane delay resulted from transmitting and processing f_l, i.e.,

$$D_l = D_l^{\mathrm{s}} + D_l^{\mathrm{c}} \tag{2}$$

where D_l^{s} is the data plane delay of f_l which is resulted from transmitting f_l through the network and processing f_l at switches, D_l^{c} represents the control plane delay caused by the information interaction between the controller and switches for inquiring and disseminating the forwarding rules of f_l.

D_l^{s} in (2) can be calculated as the transmission and processing delay of the multi-hop links along the transmission path of f_l, which is given by

$$D_l^{\mathrm{s}} = \sum_{i=1}^{N} \sum_{j=1, i\neq j}^{N} \sum_{m=1}^{M_l} y_{i,j,l}^{m} \left(D_{i,j,l}^{\mathrm{st}} + D_{i,l}^{\mathrm{sq}} \right) \tag{3}$$

where $y_{i,j,l}^{m} \in \{0, 1\}$ represents the binary flow forwarding variable of f_l. We set $y_{i,j,l}^{m} = 1$, if $E_{i,j}$ is assigned for f_l as the mth hop transmission link, otherwise, $y_{i,j,l}^{m} = 0$. M_l denotes the total hops of the transmission path of f_l from S_l to T_l, $D_{i,j,l}^{\mathrm{st}}$ denotes the transmission delay of f_l along $E_{i,j}$ and can be computed as

$$D_{i,j,l}^{\mathrm{st}} = \frac{Q_l}{\alpha_{i,j,l} B_{i,j}} \tag{4}$$

where $\alpha_{i,j,l} \in (0, 1]$ is the link resource allocation variable which represents the portion of the link capacity resource of $E_{i,j}$ being allocated to f_l.

$D_{i,l}^{\mathrm{sq}}$ in (3) represents the queueing delay of f_l at V_i. This article assumes that user flow processing at switches obeys M/M/1 queuing model, where the arrival of user requests at switches can be modeled as a Poisson stochastic process and the time required for flow processing at switches follows exponential distribution [32]. Let λ_i denote the average arrival rate of user packets at V_i, and μ_i denote the average serving rate at V_i. Since both the arrival of user flow requests and packet processing at switches change randomly, the corresponding queuing delay is very dynamic, and it is difficult to compute the instantaneous delay. For simplicity, in this article, we use average queuing delay to characterize the average waiting time of user requests in queues. Let λ_i denote the packet request rate at V_i and μ_i denote the average serving rate of V_i, the average queuing delay of f_l at V_i can be calculated as

$$D_{i,l}^{\mathrm{sq}} = \frac{1}{\beta_{i,l}(\mu_i - \lambda_i)} \tag{5}$$

where $\beta_{i,l} \in (0, 1]$ is the processing resource allocation variable which is defined as the portion of the processing capability of V_i allocated to f_l. It should be mentioned that although the parameters λ_i and μ_i are considered as constants in this article for simplicity, in practice, to obtain these parameters, we may use statistics theory or predicting methods [33]. More specifically, based on the historic information of packet requests and the processing ability of switches, we may estimate the average characteristics of random arrival of user flows and flow processing characteristics of switches. In addition, considering the dynamical features of user flow arrival and flow processing, we may also predict future characteristics of user flows and switches by applying

predicting methods, such as long short term memory (LSTM) networks [34].

While transmitting f_l from S_l to T_l, control plane delay may occur in the case that the forwarding rules of f_l have not been cached at the switches along the transmission path. The control plane delay mainly consists of the transmission and processing delay of the packet-in messages and the packet-out messages. Hence, D_l^{c} in (2) can be computed as

$$D_l^{\mathrm{c}} = D_l^{\mathrm{cs}} + \left(1 - \prod_{m=1}^{M_l} \delta_{l,m}\right)(D^{\mathrm{cq}} + D^{\mathrm{cp}}) \tag{6}$$

where D_l^{cs} represents the transmission delay of the packet-in messages and packet-out messages. In the case that one switch has received one user flow of which the forwarding rules have not been cached locally, the switch will send the packet-in messages to the controller, which designs flow forwarding strategy and disseminates the packet-out messages to the inquiring switch and its subsequent peers along the flow forwarding path, therefore, D_l^{cs} can be expressed as

$$D_l^{\mathrm{cs}} = \sum_{m=1}^{M_l} \sum_{i=1}^{N} \sum_{j=1,\, j\neq i}^{N} (1-\delta_{l,m}) \left(\sum_{m_o=m}^{M_l} y_{i,j,l}^{m_o} + y_{i,j,l}^{m}\right) \frac{S_{\mathrm{p}}}{T_{\mathrm{c},i}} \tag{7}$$

where $\delta_{l,m} \in \{0, 1\}$ denotes the binary rule caching variable of f_l, if the forwarding rules of f_l have been cached at the mth hop switch of the flow forwarding path, $\delta_{l,m} = 1$, otherwise, $\delta_{l,m} = 0$, S_{p} denotes the average size of packet-in messages and packet-out messages, $T_{\mathrm{c},i}$ denotes the link transmission rate between the controller and V_i.

D^{cq} in (6) represents the average queuing delay of the packet-in messages at the controller. Following a similar manner as computing $D_{i,l}^{\mathrm{sq}}$, we obtain

$$D^{\mathrm{cq}} = \frac{1}{\mu_{\mathrm{c}} - \lambda_{\mathrm{c}}} \tag{8}$$

where μ_{c} and λ_{c} denote the service rate and arrival rate of controller, respectively. D^{cp} in (6) represents the processing delay at the controller which is defined as the time required by the controller to calculate the forwarding strategy of one user flow, which is given by

$$D^{\mathrm{cp}} = \frac{S_{\mathrm{c}}}{F_{\mathrm{c}}} \tag{9}$$

where S_{c} denotes the amount of computation resource required to determine the flow forwarding strategy and F_{c} is the computation capability of the controller. Notice that as long as there exists one switch along the flow forwarding path of a user flow which has not cached the forwarding rules of the flow, the packet-in messages should be transmitted and processed, therefore, D^{cq} and D^{cq} in (6) are both weighted by coefficient $\left(1 - \prod_{m=1}^{M_l} \delta_{l,m}\right)$.

B. OPTIMIZATION CONSTRAINTS

Certain optimization constraints should be satisfied when designing joint rule caching and flow forwarding strategy, as discussed in detail in this subsection.

1) FLOW CONSERVATION CONSTRAINTS

While transmitting user flows with fixed traffic demand from their predefined source switches to destination switches through the network, flow conservation should be guaranteed at source switches, destination switches and intermediate switches. More specifically, if V_i is the source switch of f_l, V_i should forward f_l to one of its neighboring switches, i.e.,

$$\mathrm{C1}: \sum_{j=1,\, j\neq i}^{N} y_{i,j,l}^{1} = 1, \text{ if } V_i = \mathrm{S}_l. \tag{10}$$

Let V_j be the destination switch of f_l, it should receive f_l from one of its neighboring switches, i.e.,

$$\mathrm{C2}: \sum_{i=1,\, i\neq j}^{N} y_{i,j,l}^{M_l} = 1, \text{ if } V_j = \mathrm{T}_l. \tag{11}$$

If V_j is neither the source switch nor the destination switch of f_l, it will not create new user flows or destroy any user flows, i.e.,

$$\mathrm{C3}: \sum_{i=1, i\neq j}^{N} y_{i,j,l}^{m} = \sum_{k=1, k\neq j}^{N} y_{j,k,l}^{m+1}, \text{ if } V_j \neq \{\mathrm{S}_l, \mathrm{T}_l\}. \tag{12}$$

Note that (12) holds in two cases, i.e., Case 1: V_j is not selected to forward flow f_l, then it will not receive f_l from its neighboring switches, or send f_l to other switches. Hence, both sides of the equation equal to 0; Case 2: V_j is a relay switch which is selected to forward f_l, then it will receive f_l from one of its neighboring switches, and then send f_l to one of its neighboring switches. In this case, both sides of the equation equal to 1.

2) LINK CAPACITY RESOURCE CONSTRAINT

While it is likely that a number of user flows might be transmitted through a common link, the total amount of the link capacity utilized by all the sharing flows should meet the link capacity constraint. For link $E_{i,j}$, we obtain the following constraint:

$$\mathrm{C4}: \sum_{l=1}^{L} \sum_{m=1}^{M_l} y_{i,j,l}^{m} B_l^{\min} \leq B_{i,j}. \tag{13}$$

3) DATA RATE CONSTRAINT

To transmit user flows through the network, we should ensure that the minimum data rate requirement of the user flows is satisfied. Suppose f_l is transmitted through $E_{i,j}$, i.e., $y_{i,j,l}^{m} = 1$, the following constraint should be met:

$$\mathrm{C5}: \alpha_{i,j,l} B_{i,j} \geq B_l^{\min}, \text{ if } y_{i,j,l}^{m} = 1. \tag{14}$$

4) CACHE CAPACITY CONSTRAINT

While the TCAM of one switch may cache the forwarding rules of multiple flows, the cache capacity of the TCAM should be met. For switch V_i, the cache capacity constraint is given by

$$\text{C6}: \sum_{l=1}^{L} \sum_{m=1}^{M_l} \sum_{j=1, j\neq i}^{N} \delta_{l,m} y_{i,j,l}^{m} R_l \leq C_i. \tag{15}$$

5) RESOURCE ALLOCATION CONSTRAINTS

Resource sharing among user flows should satisfy limited resources constraints. The resources considered in this work are mainly the capacity resource of links and the processing resource of switches. Thus, the following two constraints can be obtained:

$$\text{C7}: 0 \leq \sum_{l=1}^{L} \alpha_{i,j,l} \leq 1, \tag{16}$$

$$\text{C8}: 0 \leq \sum_{l=1}^{L} \beta_{i,l} \leq 1. \tag{17}$$

6) FLOW FORWARDING CONSTRAINT

Since user flows can only be transmitted over existing physical links in the network, we obtain the following flow forwarding constraint:

$$\text{C9}: y_{i,j,l}^{m} = 0, \text{ if } x_{i,j} = 0. \tag{18}$$

C. OPTIMIZATION PROBLEM FORMULATION

The end-to-end delay minimization-based optimization problem for designing joint rule caching and flow forwarding strategy can be formulated as follows

$$\min_{\delta_{l,m}, y_{i,j,l}^{m}, \alpha_{i,j,l}, \beta_{i,l}} \quad D$$
$$\text{s.t. C1} - \text{C9}. \tag{19}$$

V. SOLUTION TO THE OPTIMIZATION PROBLEM

The optimization problem formulated in (19) involves the joint optimization of rule caching and flow forwarding strategy. Under the assumption that the rule caching strategy is given, the problem of flow forwarding in an SDN scenario can be regarded as the classical traveling salesman problem (TSP) which has been proved to be an NP-hard problem [35]. Hence, the optimization problem in (19) is also NP-hard, which cannot be solved directly.

In this section, to solve the formulated optimization problem, we transform the problem into three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem, and resource sharing subproblem. By solving the three subproblems successively, the joint rule caching and flow forwarding strategy is obtained.

A. FLOW FORWARDING SUBPROBLEM FOR INDIVIDUAL USER FLOWS

It is apparent that for individual user flows, caching flow forwarding rules at the switches along their transmission paths is highly desired, in this subsection, we first consider an ideal case, i.e., the forwarding rules of all user flows have been cached at the TCAM of switches along the flow forwarding paths. Under this assumption, we are able to remove the constraints of rule caching, and therefore, the formulated joint rule caching and flow forwarding problem is reduced to a flow forwarding subproblem.

1) SUBPROBLEM FORMULATION

Under the given rule caching assumption, i.e., $\delta_{l,m} = 1$, $\forall\ l, m$, the control plane delay is negligible in counting the end-to-end delay and the overall delay of all user flows in the network consists of data plane delay only, i.e., $D = \sum_{l=1}^{L} D_l^{\text{s}}$, therefore, the optimization problem formulated in (19) is reduced to the following flow forwarding subproblem:

$$\min_{y_{i,j,l}^{m}, \alpha_{i,j,l}, \beta_{i,l}} \sum_{l=1}^{L} D_l^{\text{s}}$$
$$\text{s.t. C1} - \text{C5},\ \text{C7} - \text{C9 in (19)}. \tag{20}$$

The above optimization problem is still difficult to solve mainly due to the coupling and resource sharing among multiple user flows. To tackle this problem, we first consider the optimal flow forwarding strategy of individual user flows and then deal with resource sharing issues of user flows.

Aiming to design the optimal flow forwarding strategy for an individual user flow, say f_l, we assume that no resource sharing exists among various user flows. More specifically, if $E_{i,j}$ is selected to transmit f_l, all the link resource of $E_{i,j}$ and the processing resource of V_i will be allocated to f_l, hence, we obtain $\alpha_{i,j,l} = 1$, $\beta_{i,l} = 1$. Let $D_l^{(1)}$ be the data plane delay of f_l which is computed as the sum of the transmission and queuing delay, we may express $D_l^{(1)}$ as

$$D_l^{(1)} = \sum_{m=1}^{M_l} \sum_{i=1}^{N} \sum_{j=1, i\neq j}^{N} y_{i,j,l}^{m} \left(\frac{Q_l}{B_{i,j}} + \frac{1}{\mu_i - \lambda_i} \right). \tag{21}$$

The flow forwarding subproblem of f_l which minimizes $D_l^{(1)}$ can be formulated as

$$\min_{y_{i,j,l}^{m}} \quad D_l^{(1)}$$
$$\text{s.t.} \quad \text{C1} - \text{C4},\ \text{C9 in (19)}. \tag{22}$$

2) K-SHORTEST PATH ALGORITHM-BASED FLOW FORWARDING STRATEGY

Through solving the problem formulated in (22), we will be able to obtain the optimal flow forwarding strategy of f_l. However, it should be noticed that the obtained flow forwarding strategy only achieves data plane delay minimization, i.e., the

minimum $D_l^{(1)}$. On account of the correlation between flow forwarding, rule caching and resource sharing, the obtained flow forwarding strategy may not result in the optimization of the total end-to-end delay which is composed of both data plane delay and control plane delay.

In this article, stressing the trade-off between transmission performance and computational complexity, we propose a flow forwarding strategy which selects multiple optimal candidate flow forwarding paths instead of one path. Then, jointly considering the network performance affected by rule caching and resource sharing strategy, the candidate path offering the minimum total end-to-end delay is selected. To solve the optimization problem formulated in (22) and obtain multiple candidate paths, we apply K-shortest path algorithm which is capable of choosing K shortest paths between two nodes in a weighted graph [36].

Introducing link weight set W to the SDN network graph $G = (V, E)$, we obtain a weighted graph $G = (V, E, W)$, where $W = \{W_{i,j}, 1 \le i, j \le N, i \ne j\}$ denotes the set of link weight, and $W_{i,j}$ is the weight of $E_{i,j}$, which is given by

$$W_{i,j} = \frac{Q_l}{B_{i,j}} + \frac{1}{\mu_i - \lambda_i}. \tag{23}$$

Given the network weighted topology graph $G = (V, E, W)$, we will be able to find out K candidate flow forwarding paths with the minimum $D_l^{(1)}$ by utilizing the K-shortest path algorithm.

As an extension of the Dijkstra algorithm, the K-shortest path algorithm employs the idea of deviating path on the path selection strategy obtained based on the Dijkstra algorithm [37]. The algorithm for determining the K shortest paths for f_l can be described briefly as follows. Applying the Dijkstra algorithm, we determine the shortest forwarding path between S_l and T_l. Let $p_l^{(1)}$ denote the first shortest path for forwarding f_l. Then, removing the individual links along the shortest path $p_l^{(1)}$ sequentially, we reselect the shortest forwarding path between S_l and T_l. Repeat the above process until the K shortest forwarding paths are obtained.

The steps of the K-shortest path algorithm-based flow forwarding procedure are summarized as follows:

1) For user flow f_l, set $k = 1$.
2) Characterize the considered network model by a weighted topology graph $G = (V, E, W)$, where $W = \{W_{i,j}\}$, $W_{i,j} = \frac{Q_l}{B_{i,j}} + \frac{1}{\mu_i - \lambda_i}$.
3) Obtain the shortest path from S_l to T_l by applying the Dijkstra algorithm, denoted by $p_l^{(k)}$.
4) Remove the individual links on $p_l^{(k)}$ sequentially, update the weighted graph G, and set $k = k + 1$.
5) Reselect the shortest path from S_l to T_l by applying the Dijkstra algorithm, denoted by $p_l^{(k)}$.
6) Repeat Step 4 and Step 5, until $k = K$.

Let $P_l = \left\{p_l^{(1)}, p_l^{(2)}, \ldots, p_l^{(K)}\right\}$ denote the obtained K shortest path set of f_l. We further denote $y_{i,j,l}^{(m,k)}$ as the mth hop forwarding strategy of the kth candidate path of f_l, and $D_l^{(1,k)}$ as the data plane delay of the kth candidate path of f_l, $1 \le m \le M_l$, $1 \le k \le K$, $1 \le l \le L$.

B. RULE CACHING AND CANDIDATE PATH SELECTION SUBPROBLEM

According to the solution of the flow forwarding subproblem in (22), we obtain the K candidate forwarding paths for each user flow. It should be mentioned that the K candidate forwarding paths are determined for individual user flows under the ideal rule caching assumption, however, given the rule caching constraint C6 formulated in (15), the assumption may not hold for all the switches, in which case, the feasible rule caching strategy meeting the constraint should be designed.

Apparently, if the performance of the K-shortest paths can be further examined easily by considering the rule caching strategy, we should select the best one among K paths based on the constraints mentioned in Section IV, however, due to the caching resource competition among user flows, it is impossible to compute the performance of individual paths independently. In particular, given various rule caching strategies, the performance of the K-shortest paths may change. Although we may extensively search all possible rule caching strategies and compute the performance of the K paths, the computational complexity can be prohibited. In this article, given the flow forwarding strategy obtained for all user flows, we formulate rule caching and candidate path selection subproblem and obtain the joint strategy by solving the subproblem.

1) SUBPROBLEM FORMULATION

Let $D_l^{(2,k)}$ denote the control plane delay of f_l when choosing the kth candidate path, we may express $D_l^{(2,k)}$ as

$$D_l^{(2,k)} = \left(1 - \prod_{m=1}^{M_l} \delta_{l,m}^{(k)}\right)\left(\frac{1}{\mu_c - \lambda_c} + \frac{S_c}{F_c}\right) + \sum_{m=1}^{M_l}\sum_{i=1}^{N}\sum_{j=1,\, j\ne i}^{N}\left(1 - \delta_{l,m}^{(k)}\right)\left(\sum_{m_o=m}^{M_l} y_{i,j,l}^{(m_o,k)} + y_{i,j,l}^{(m,k)}\right)\frac{S_p}{T_{c,i}} \tag{24}$$

where $\delta_{l,m}^{(k)}$ denotes the rule caching strategy of the mth hop switch on the kth candidate path of f_l. Since different user flows may choose various candidate flow forwarding paths, to characterize the path selection strategy of user flows, we introduce a binary candidate path selection variable $\gamma_l^{(k)}$. We set $\gamma_l^{(k)} = 1$, if f_l selects the kth candidate path $p_l^{(k)}$; otherwise, $\gamma_l^{(k)} = 0$.

Let $\bar{D}$ denote the total end-to-end delay of user flows given K candidate path strategy, we express $\bar{D}$ as

$$\bar{D} = \sum_{l=1}^{L}\sum_{k=1}^{K}\gamma_l^{(k)}\left(D_l^{(1,k)} + D_l^{(2,k)}\right). \tag{25}$$

The rule caching and candidate path selection subproblem of user flows can be formulated as

$$\begin{aligned} \min_{\delta_{l,m}^{(k)},\ \gamma_l^{(k)}} \quad & \bar{D} \\ \text{s.t.} \quad & \sum_{l=1}^{L}\sum_{m=1}^{M_l}\sum_{j=1,j\neq i}^{N} \delta_{l,m}^{(k)} y_{i,j,l}^{(m,k)} R_l \leq C_i \\ & \sum_{k=1}^{K} \gamma_l^{(k)} = 1. \end{aligned} \tag{26}$$

where the constraint $\sum_{k=1}^{K} \gamma_l^{(k)} = 1$ indicates that each user flow can only select one candidate path.

It is obvious that given K candidate paths of L user flows, we may obtain a large number of possibilities which assign different user flows to various candidate paths. However, among the K candidate paths of individual user flows, we may tend to assign the first candidate path to user flows since the first candidate path offers the shortest data plane delay. As a result, to jointly design rule caching and candidate path selection strategy, it is reasonable and straightforward to start from examining the rule cache space constraint of the switches along the first candidate path of user flows, and designing the corresponding rule caching and candidate path selection strategy. In the case that the rule caching and resource sharing constraints of the first candidate paths of user flows cannot be met, we should assign other candidate paths to user flows and design rule caching strategy accordingly.

According to the rule cache space status of the switches along the first candidate path of all user flows, we may obtain the following two cases, i.e., Case 1: sufficient rule caching space, and Case 2: insufficient rule caching space.

2) RULE CACHING STRATEGY FOR CASE 1: SUFFICIENT RULE CACHING SPACE

We first consider the situation where the cache space resources of the switches along the first candidate path of user flows are sufficient to cache the flow forwarding rules. In this case, we assign the first candidate paths to all user flows and cache flow forwarding rules at the switches accordingly. More specifically, let $\gamma_l^{(k,*)}$ denote the optimal candidate path selection strategy of f_l and $\delta_{l,m}^{(k,*)}$ denote the optimal rule caching strategy of the mth hop switch on the kth candidate path of f_l, we set $\gamma_l^{(1,*)} = 1$ and $\delta_{l,m}^{(1,*)} = 1$. For $k \neq 1$, we set $\gamma_l^{(k,*)} = 0$ and $\delta_{l,m}^{(k,*)} = 0$, $1 \leq l \leq L$.

3) RULE CACHING STRATEGY FOR CASE 2: INSUFFICIENT RULE CACHING SPACE

While caching flow forwarding rules at the switches along the first candidate path of user flows is highly desired, the limited TCAM storage capacity of switches may result in the failure to cache the flow forwarding rules of user flows. In this case, cache conflict occurs at the switches. To tackle this problem and design the feasible rule caching and candidate path selection strategy, we propose a priority-based rule caching algorithm. More specifically, for the common switches shared by a certain number of user flows, we examine the hops of the shared switches, and assign the highest priority to the user flow of which the shared switch has the smallest hop. The rationality of this idea is that caching flow forwarding rules at the switches with small hops leads to smaller control plane delay and least waste of the caching space of subsequent switches [6], [9].

We denote Z as the number of user flows of which the first candidate path share common switches. Let $F_0 = \left\{f_{l_1}, \cdots, f_{l_z}, \cdots, f_{l_Z}\right\}, 1 \leq z \leq Z$, denote the set of user flows where the shared switches exist, $V_0 = \left\{V_{l_1}, \cdots, V_{l_a}, \cdots, V_{l_A}\right\}, 1 \leq a \leq A$, denote the set of shared switches. Further denote $h_{a,z}$ as the number of hops of V_{l_a} in the first candidate path of f_{l_z}. In the case that V_{l_a} is not the shared switch of f_{l_z}, we set $h_{a,z} = \infty$. The priority of the shared user flows is determined according to the value of $h_{a,z}$. Specifically, if switch $V_{l_{a^*}}$ on user flow $f_{l_{z^*}}$ is of the minimum value of $h_{a,z}$, i.e.,

$$\left\{a^*, z^*\right\} = \arg\min \left\{h_{a,z}\right\}, \tag{27}$$

flow $f_{l_z}^*$ is assigned the highest priority. Removing $f_{l_z}^*$ from the shared flows and repeat the above process until the priority of all the shared user flows is determined. In addition, in the case that the forwarding hops of the switches on multiple flows is equal, the priority of the flows can be further determined based on the number of shared switches. Specifically, the flows with a smaller number of shared switches will be assigned a higher priority. Table 1 shows an example of the forwarding hops of switches of multiple user flows.

TABLE 1. The forwarding hops of shared flows and shared switches.

	f_{l_1}	f_{l_2}	...	f_{l_z}	...	f_{l_Z}
V_{l_1}	$h_{1,1}$	$h_{1,2}$	...	∞	...	$h_{1,Z}$
V_{l_2}	∞	$h_{2,2}$	...	$h_{2,z}$	...	$h_{2,Z}$
...	...	...	...	...	...	...
V_{l_a}	$h_{a,1}$	∞	...	$h_{a,z}$	...	∞
...	...	...	...	...	...	...
V_{l_A}	$h_{A,1}$	$h_{A,2}$	...	∞	...	$h_{A,Z}$

Without loss of generality, we denote S_{l_z} as the priority of f_{l_z}, where $S_{l_1} \geq S_{l_2} \geq \ldots \geq S_{l_Z}$. Given the priority of conflicting user flows, the rule caching strategy can be designed succesively and candidate path selection strategy can be determined accordingly. Under the assumption that user flow f_{l_z} is of the highest priority, the proposed priority-based rule caching algorithm is described as follows.

1) For user flow f_{l_z}, check whether V_{l_a} is shared by f_{l_z} and other user flows, $1 \leq z \leq Z$, $1 \leq a \leq A$.
2) If yes, check whether the caching space of V_{l_a} is sufficient for storing the forwarding rule of f_{l_z}.
3) If the cache space of V_{l_a} is sufficient, caching the forwarding rule at V_{l_a}, i.e, $\delta_{l_z,h_{a,z}}^{(1,*)} = 1$.
4) If the cache space of V_{l_a} is insufficient, it is infeasible to cache the forwarding rule of f_{l_z} at V_{l_a} and its succeeding

switches along the first candidate path. Thus, we set $\delta_{l_z,m}^{(1,*)} = 0$, $m = h_{a,z}, h_{a,z}+1, \ldots, M_{l_z}$, where M_{l_z} denotes the maximum hop of the first candidate path of f_{l_z}.

5) Substituting $\delta_{l_z,m}^{(1,*)}$ to $D_{l_z}^{(1,1)} + D_{l_z}^{(2,1)}$, and following a similar manner, design rule caching strategy for the remaining candidate paths of f_{l_z}, and obtain $D_{l_z}^{(1,k)} + D_{l_z}^{(2,k)}$, $2 \le k \le K$.
6) The optimal forwarding path $p_{l_z}^{(k,*)}$ is selected for f_{l_z}, where $p_{l_z}^{(k,*)} = \arg\min \left\{ D_{l_z}^{(1,k)} + D_{l_z}^{(2,k)} \right\}$, $1 \le k \le K$, and the corresponding rule caching strategy is the optimal strategy.
7) Update the caching space of V_{l_a} and delete f_{l_z} from the set of user flows which share common switches.
8) Repeat the above steps for all user flows sharing common switches based on the priority of user flows.

C. RESOURCE SHARING SUBPROBLEM

Since a number of user flows might be transmitted through the network simultaneously, it is likely that multiple user flows may arrive at the same switches or might be transmitted along the same links. In this case, resource sharing strategy has to be designed. For convenience, we assume that multiple user flows will be transmitted through V_i and $E_{i,j}$ simultaneously, i.e., $\sum_{l=1}^{L}\sum_{m=1}^{M_l}\sum_{j=1,\, j\neq i}^{N} y_{i,j,l}^{(m,*)} > 1$ and $\sum_{l=1}^{L}\sum_{m=1}^{M_l} y_{i,j,l}^{(m,*)} > 1$.

The data plane delay of the user flows sharing V_i and $E_{i,j}$ can be calculated as

$$D_{i,j} = \sum_{l=1}^{L}\left(y_{i,j,l}^{(m,*)} \frac{Q_l}{\alpha_{i,j,l} B_{i,j}} + \sum_{j=1,j\neq i}^{N} y_{i,j,l}^{(m,*)} \frac{1}{\beta_{i,l}\left(\mu_i - \lambda_i\right)} \right). \tag{28}$$

The resource sharing subproblem is expressed as follows

$$\begin{aligned} \min_{\alpha_{i,j,l},\ \beta_{i,l}} \quad & D_{i,j} \\ \text{s.t.} \quad & \alpha_{i,j,l} B_{i,j} \ge B_l^{\min} \\ & 0 < \sum_{l=1}^{L} \alpha_{i,j,l} \le 1 \\ & 0 < \sum_{l=1}^{L} \beta_{i,l} \le 1 \\ & 0 < \alpha_{i,j,l} \le 1 \\ & 0 < \beta_{i,l} \le 1. \end{aligned} \tag{29}$$

It can be demonstrated that the above resource sharing subproblem is a convex problem with linear constraints, hence, can be solved by optimization tools, such as the Lagrange dual method. The corresponding Lagrange function of the problem formulated in (29) is given by

$$\begin{aligned} & L\left(\eta_{i,j,l}, \varphi_{i,l}, \omega_{i,j}, \pi_i, \alpha_{i,j,l}, \beta_{i,l}\right) \\ & = D_{i,j} + \sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \eta_{i,j,l}\left(\alpha_{i,j,l} - 1\right) + \sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \varphi_{i,l}\left(\beta_{i,l} - 1\right) \\ & \quad + \omega_{i,j}\Big(\sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \alpha_{i,j,l} - 1 \Big) + \pi_i \Big(\sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \beta_{i,l} - 1 \Big) \end{aligned} \tag{30}$$

where $\eta_{i,j,l}$, $\varphi_{i,l}$, $\omega_{i,j}$, π_i are Lagrangian multipliers. The Lagrange dual problem of the optimization problem in (29) can be formulated as

$$\begin{aligned} \max_{\eta_{i,j,l},\varphi_{i,l},\omega_{i,j},\pi_i} \min_{\alpha_{i,j,l},\beta_{i,l}} \quad & L(\eta_{i,j,l}, \varphi_{i,l}, \omega_{i,j}, \pi_i, \alpha_{i,j,l}, \beta_{i,l}) \\ \text{s.t.} \quad & \eta_{i,j,l}, \varphi_{i,l}, \omega_{i,j}, \pi_i \ge 0. \end{aligned} \tag{31}$$

For a given set of Lagrangian multipliers $\eta_{i,j,l}$, $\varphi_{i,l}$, $\omega_{i,j}$, π_i, the optimal resource allocation ratios of the shared switch and link can be respectively obtained as

$$\alpha_{i,j,l}^{*} = \left[\sqrt{\frac{Q_l}{B_{i,j}\left(\eta_{i,j,l} + \omega_{i,j}\right)}} \right]^{+}, \tag{32}$$

$$\beta_{i,l}^{*} = \left[\sqrt{\frac{1}{\left(\mu_i - \lambda_i\right)\left(\varphi_{i,l} + \pi_i\right)}} \right]^{+}. \tag{33}$$

By applying the gradient method, the Lagrangian multipliers can be updated as

$$\eta_{i,j,l}(t+1) = \left[\eta_{i,j,l}(t) + \theta_1\left(\alpha_{i,j,l} - 1\right)\right]^{+}, \tag{34}$$

$$\varphi_{i,l}(t+1) = \left[\varphi_{i,l}(t) + \theta_2\left(\beta_{i,l} - 1\right)\right]^{+}, \tag{35}$$

$$\omega_{i,j}(t+1) = \left[\omega_{i,j}(t) + \theta_3 \left(\sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \alpha_{i,j,l} - 1 \right) \right]^{+}, \tag{36}$$

$$\pi_i(t+1) = \left[\pi_i(t) + \theta_4 \left(\sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \beta_{i,l} - 1 \right) \right]^{+}. \tag{37}$$

where $\theta_k, k = 1, 2, 3, 4$ is the positive step size.

The pseudo-code of the proposed algorithm for multiple user flows is described in Algorithm 1.

VI. COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of our proposed algorithm. Since the three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem and resource sharing subproblem are successively solved, we examine the complexity of solving the three subprolbems, respectively.

A. FLOW FORWARDING SUBPROBLEM

To solve the formulated flow forwarding subproblem, we determine K shortest forwarding paths for all user flows. It can be proved that the required computational complexity for finding one shortest path is $O(N(Y + N\log N))$, where Y denotes the number of links in the network. As we should determine K candidate forwarding paths for L user flows, the computational complexity is $O(KNL(Y + N\log N))$.

Algorithm 1 Proposed Joint Rule Caching and Flow Forwarding Algorithm

A network weighted topology graph $G = (V, E, W)$ is created
for $l = 1 : L$ **do**
  On graph G, run the K-shortest path algorithm
  Obtain K candidate paths $P_l = \{P_l^1, P_l^2, \ldots, P_l^K\}$ for f_l, let $y_{i,j,l}^{m,k}$ denote the flow forwarding strategy corresponding to the kth candidate path of f_l.
end for
if The TCAM flow table in the switches has sufficient rule cache space **then**
  Set the optimal rule caching strategy $\delta_{l,m}^* = 1, \forall m$
else
  if existing shared switches and links **then**
    Define the priority of user flows at the shared switches
    if shared switches have insufficient rule cache space **then**
      Apply the proposed priority-based heuristic rule caching and flow forwarding algorithm
      Obtain the flow forwarding strategy $P_{l_z}^{k^*} = \arg\min \left\{D_{l_z}^{2,k}\right\}, 1 \le k \le K$, and the corresponding rule caching strategy.
    end if
    if $\sum_{l=1}^{L} y_{i,j,l}^{m} > 1$ **then**
      $D_{i,j} = \sum_{l=1, y_{i,j,l}^{m,*}=1}^{L} \frac{Q_l}{\alpha_{i,j,l} B_{i,j}}$,
      Obtain $\alpha_{i,j,l}^* = \arg\min D_{i,j}$ by applying Lagrange dual method
    end if
    Similarly, $\beta_{i,l}^*$ can be obtained by applying Lagrange dual method
  else
    $\alpha_{i,j,l}^* = 1, \beta_{i,l}^* = 1$
  end if
end if

B. RULE CACHING AND CANDIDATE PATH SELECTION SUBPROBLEM

To solve the formulated rule caching and candidate path selection subproblem, we consider two cases, i.e., Case 1: sufficient rule caching space, and Case 2: insufficient rule caching space, and design the rule caching strategies accordingly.

For Case 1, under the assumption that the cache space of switches along the first candidate path of all user flows is sufficient, we assign the first candidate path to all user flows and cache the flow forwarding rules at the switches along the paths. Let $M_{l,1}$ denote the number of hops of the first candidate hop of f_l, the computational complexity can be calculated as $L \max\{M_{l,1}\}$.

For Case 2, we propose a priority-based rule caching algorithm for determining the rule caching and candidate path selection strategy. Let Z denote the number of conflicting user flows, and A denote the average number of shared switches of conflicting flows, to determine the priority of conflicting user flows, the required computational complexity is $O(AZ)$. Since rule caching strategy should be determined for all the shared switches of the conflicting flows, and the performance of K candidate paths should be examined and compared, the required computational complexity is $O(AZK)$. As a result, to solve the rule caching and candidate path selection subproblem, the overall complexity is $O(AZK)$.

C. RESOURCE SHARING SUBPROBLEM

To examine the computational complexity of the formulated resource sharing subproblem, we denote Z_1 and Z_2 as the number of shared switches and links, respectively, X_1 and X_2 as the number of user flows sharing switches and links, respectively. As Lagrange multipliers should be updated for each shared switch and link, the computational complexity for executing the resource sharing algorithm is $O(K_0(Z_1X_1 + Z_2X_2))$, where K_0 denotes the number of iterations required for the algorithm to achieve convergence.

VII. SIMULATION RESULTS

In this section, we evaluate the performance of the proposed joint scheme and that of previously proposed solutions through exhaustive numerical simulations. The simulation tool used in this article is Matlab software. In the simulation, we set up an SDN network where switches are randomly located within an square region of 400m×400m and links between any two switches are generated randomly. In addition, we consider network scenarios with different numbers of user flows and forwarding rules.

To characterize various requirements of user flows in the simulation, simulation parameters such as traffic demand, the minimum transmission rate requirement, link capacity, TCAM size and the transmission rate between switches are randomly selected from a certain set. Unless otherwise stated, the parameters employed in the simulations are shown in Table 2. Simulation results are counted based on 500 experimental outcomes resulted from randomly chosen simulation parameters.

Figure 2 compares the total delay resulted by our proposed algorithm and the schemes proposed in [20] and [28] under the scenarios where the traffic demand of user flows ranges from 10 to 50. As shown in (4), higher traffic demand of user flows requires longer transmission delay, resulting in longer total delay, which is demonstrated in the figure. On the other hand, we can see that as the number of user flows increases, the total end-to-end delay also increases. This is because we compute the delay of all user flows as the summation of the delay of individual user flows, as defined in (1). The curves in Figure 2 show that the proposed solution provides lower delay than the schemes proposed in [20] and [28]. This is mainly because the algorithm proposed in this article

TABLE 2. Simulation parameters.

Parameters	Value
Traffic demand (Q_l)	[5, 50]Mbit
Link capacity ($B_{i,j}$)	[80, 200]Mbps
Transmission rate ($T_{c,i}$)	[1, 10]Mbps
TCAM size of V_i (C_i)	[100, 500]
Number of forwarding rules (R_l)	[50, 100]
Service rate of V_i (μ_i)	[20, 30]
Arrival rate of V_i (λ_i)	[1, 9]
Service rate of controller (μ_c)	[100, 150]
Arrival rate of controller (λ_c)	[1, 50]
Size of packet-in/packet-out messages (S_p)	[100, 200]bytes
Required amount of computation resource (S_c)	[100, 200]Megacycles
Computation capability of controller (F_c)	[5, 8]Gigacycles/s

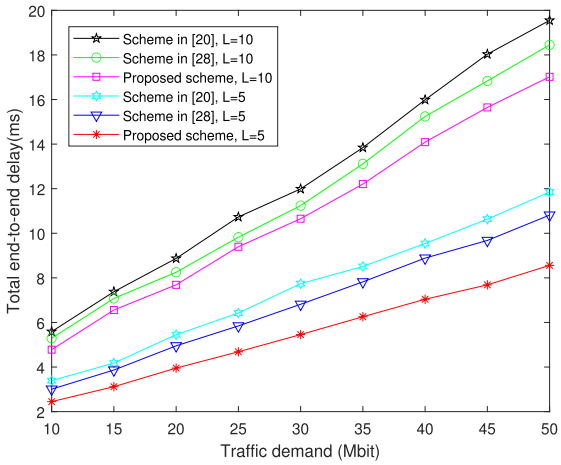

FIGURE 2. Total end-to-end delay versus traffic demand (different number of user flows).

jointly considers rule caching and flow forwarding issues and focuses on minimizing the total data plane and control plane delay of user flows, while the scheme proposed in [20] only investigates flow forwarding strategy and fails to consider rule caching strategy, thus may lead to undesired control plane latency, and the algorithm proposed in [28] aims to reduce the space occupation of flow forwarding rules, which might lead to excessively long delay.

The impacts of traffic demands of user flows on total end-to-end delay are shown in Figure 3. The results are obtained based on our proposed algorithm and the algorithms proposed in [20] and [28]. Observing the curves in the figure, we can see that higher traffic demands of user flows result in longer end-to-end delay. This is mainly because transmitting larger amount of user flows requires longer transmission delay, as clearly shown in (4). In the figure, we also examine the effect of the number of forwarding rules on network performance. The total end-to-end delay of user flows is plotted when the number of forwarding rules is chosen as $R = 50$ and $R = 100$, respectively. We can observe from the figure that a larger number of forwarding rules results in longer total end-to-end delay. This is because according to constraint C6 in (15), the rule caching capacity should be met at switches. Given fixed caching capacity at switches,

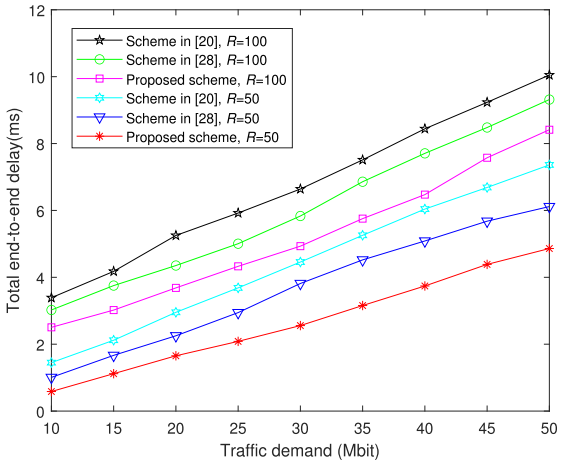

FIGURE 3. Total end-to-end delay versus traffic demand (different number of rules).

if user flows require a larger number of forwarding rules, the number of user flows whose forwarding rules can be cached will decrease. As a result, the controller needs to determine flow forwarding strategy for a larger number of user flows, causing longer control plane delay and total end-to-end delay as well. Furthermore, the figure also shows that our proposed algorithm outperforms the two comparative schemes, i.e., the schemes proposed in [20] and [28], and the advantage becomes obvious with the number of forwarding rules increased.

The total delay of user flows obtained from different network scales is plotted in Figure 4, where network scale is characterized by different number of switches, varying from 40 to 100. The figure shows a larger network scale results in enhanced performance in end-to-end delay. The reason is that a larger network scale provides an increased number of available switches and links in the network, thus offering higher flexibility in selecting flow forwarding paths and rule caching switches, and resulting in improved end-to-end delay performance in turn. We can also see that as the number

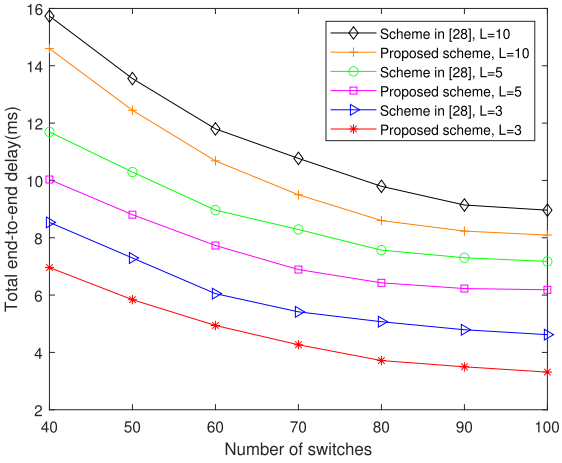

FIGURE 4. Total end-to-end delay versus the number of switches (different number of user flows).

of user flows increases, the total end-to-end delay increases accordingly. Obviously, we see that the proposed solution incurs less delay compared to the algorithm proposed in [28]. In particular, comparing the total end-to-end delay obtained from our proposed algorithm and the algorithm proposed in [28], we can observe that our proposed scheme is capable of reducing end-to-end delay approximately by 20%.

In Figure 5, we vary the TCAM capacity of switches from 100 to 500 and plot the corresponding end-to-end delay. We can see that as TCAM capacity increases, the total end-to-end delay becomes lower. Analyzing the reason behind, we can understand that constrained by C6 in (15), switches with higher TCAM capacity are capable of caching larger number of forwarding rules. That is, the forwarding strategy of a larger number of user flows can be cached at switches, hence, less control plane interaction is required, leading to lower control plane delay and end-to-end delay as well.

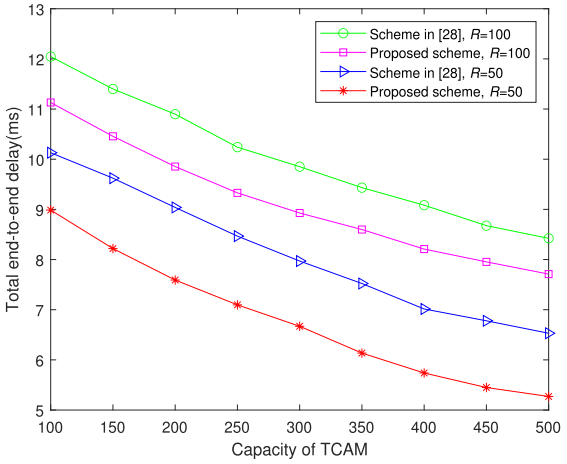

FIGURE 5. Total end-to-end delay versus capacity of TCAM (different number of rules).

To plot the above figures, the computing time or the simulation time is mainly determined by the factors such as network scale, the number of user flows and required flow forwarding rules, etc. In the case that the number of user flows is 10, the number of switches is 45 and the number of required flows forwarding rules of each user flow is 100, the average computation time for our proposed algorithm for all user flows is about 3.5 seconds for one random parameter setting. Since the algorithm proposed in [20] used randomized rounding for route selection, and derived an integer solution for each user flow to appoint a feasible path rather than to determine the K candidate paths, the average computation time is slightly lower than our algorithm, which is less than 3 seconds every time. In [28], the algorithm was proposed to solve the rule placement problem without considering the given candidate paths, the average computation time is also slightly shorter than our proposed algorithm.

We now examine the traffic overhead of our proposed approach and the existing ones. It should be noted that our proposed algorithm is a centralized scheme, which is designed by the controller based on its collected information, including the service characteristics of user flows and the transmission performance of switches. To send the service characteristics of user flow f_l to the controller, the source switch S_l or one of the intermediate switches of f_l should send a packet-in message to controller, which contains the traffic demand of f_l denoted by Q_l and the number of forwarding rules of f_l denoted by R_l, and the required traffic load is $\frac{1}{2}LM_l$. In addition, switch V_i should also send its status information to the controllers, including its average flow arrival rate λ_i, average serving rate μ_i, TCAM storage size C_i and the capacity of $E_{i,j}$, denoted by $B_{i,j}$, and the required traffic load is $\frac{LN^2}{2}$. Jointly considering the traffic load required for the controller to collect the service characteristics of user flows and the status information of switches, we obtain the overall traffic load can be computed as $\frac{1}{2}L(M_l + N^2)$.

In [20], to determine the routes of all flows, the header packet of each new-arrival flow will be reported to the controller and the controller will collect the size of each flow from switches. Let L denote the number of user flows and N denote the number of switches, the required traffic load is LN^2. In order to update route scheduling with delay constraint for all flows, the algorithm should be repeated until the update delay is running out, and the controller will collect the information of each flow during each iteration. Let K_0 denote the number of iterations, the required traffic load is LK_0. Therefore, the overall traffic load in [20] can be computed as $L(N^2 + K_0)$. In [28], in order to minimize the rule space occupation, the flow caching rules are decomposed into multiple subsets, and can be placed along the routing paths in an arbitrary order. Let K_1 denote the number of unicast sessions, and N denote the number of network devices, the required traffic load is NK_1.

VIII. DISCUSSION AND LIMITATION

In this article, we jointly design rule caching and flow forwarding strategy for user flows in SDN. Considering the importance of end-to-end delay of user flows, we formulate the joint optimization problem as an end-to-end delay minimization problem and present a heuristic algorithm to solve it and obtain the joint rule caching and flow forwarding strategy.

Unlike previous research work, the problems of rule caching and flow forwarding are jointly considered and the total end-to-end delay is optimized in this article. Due to the close correlation between the two issues, it is difficult to solve the formulated joint rule caching and flow forwarding problem. We transform the original formulated problem into three subproblems, i.e., flow forwarding subproblem, rule caching and candidate path selection subproblem and resource sharing subproblem. Considering the impacts of rule caching and resource sharing on network performance, to solve the flow forwarding subproblem, instead of selecting one optimal forwarding path for individual user flows, we apply the K-shortest path algorithm and find K candidate paths for each user flow. Among K candidate paths, rule

caching strategy can be determined and the optimal candidate path is selected accordingly.

In the case that rule cache capacity of switches can not be satisfied and cache conflict occurs at switches, designing corresponding rule caching strategy is challenging. To tackle the problem, we propose a priority-based rule caching algorithm and design rule caching strategy for conflicting user flows accordingly to their priority. By assigning the highest priority to the user flow of which the shared switch has the smallest hop, control plane delay reduction is achieved and efficient utilization of the caching space of switches can be obtained. Through simulation experiments, we demonstrate that our proposed algorithm outperforms previously proposed algorithms.

It should be mentioned that although our proposed algorithm is of certain novelty and contributions, it also has some limitations and can be extended in future work. For example, similar as most of research work stressing flow forwarding, we consider a quasi-static scenario where the characteristics and requirements of user flows are pre-defined and the network scenario is fixed during the process of flow forwarding. In our future work, we may extend the network model and flow forwarding requirements to a dynamic scenario where user flows may arrive and depart randomly. Since random variables and random processes should be handled in this case, we may apply random graph theory or random optimization model to solve the dynamic rule caching and flow forwarding strategy.

Furthermore, in this work, we stress algorithm design and make relatively simple assumptions on system model, especially the control plane model. In particular, we assume that the control layer of SDN only contains one SDN controller which is responsible for monitoring network states and determining network management strategies. In our future work, we may extend current network model into a hierarchical controller architecture, which consists of one main controller and a number of lower-layer controllers. In this case, signaling interaction among controllers and the association between controllers and switches should be considered when designing flow forwarding strategy. We may also refer to [27] and extend the assumption of our system model by allowing certain authority switches to be able to design flow forwarding strategy, in which case the reasonable selection strategy of authority switches should be designed.

In Section VI, we have examined the computational complexity of our proposed algorithm. It can be seen from the result that the computational complexity highly depends on network scale and the number of user flows. In the case of small network scale, say, small N, Y, and small number of conflicting user flows and switches, i.e., small A, Z, the computational complexity of our proposed algorithm is relatively small. However, in a network scenario consisting a large number of switches, or a large number of user flows, the resulted computational complexity can be relatively high. While selecting a suitable number of candidate paths for flow forwarding is of importance in achieving the trade-off between computational complexity and performance enhancement, we may also explore alternative schemes which offer acceptable network performance with relatively low computational complexity.

IX. CONCLUSION

In this work, the problem of joint rule caching and flow forwarding for multiple user flows in SDN was investigated. To emphasize the importance of the end-to-end delay caused by the transmission and processing delay of data plane and control plane, we formulated the joint optimization problem as an end-to-end delay minimization problem. As the original optimization problem is an NP-hard problem, which cannot be solved directly, we proposed a heuristic algorithm which transforms the optimization problem into flow forwarding subproblem, rule caching and candidate path selection subproblem, and resource sharing subproblem. Applying the K-shortest path algorithm, a priority-based rule caching algorithm and the Lagrangian dual method to solve the three subproblems, we obtained the joint rule caching and flow forwarding strategy. The simulation experiments were conducted which show that our algorithm significantly outperforms the previous solutions.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015.
[3] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, pp. 24290–24307, 2019.
[4] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 464–486, 1st Quart., 2019.
[5] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 472–503, 1st Quart., 2020.
[6] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable OpenFlow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107346–107379, 2019.
[7] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. Vasilakos, "Survey on routing in data centers: Insights and future directions," *IEEE Netw.*, vol. 25, no. 4, pp. 6–10, Jul. 2011.
[8] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 388–415, 1st Quart., 2018.
[9] H. Huang, S. Guo, P. Li, W. Liang, and A. Y. Zomaya, "Cost minimization for rule caching in software defined networking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1007–1016, Apr. 2016.
[10] Z. Ding, X. Fan, J. Yu, and J. Bi, "Update cost-aware cache replacement for wildcard rules in software-defined networking," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 457–463.
[11] D. Wang, Q. Li, Y. Jiang, M. Xu, and G. Hu, "Balancer: A traffic-aware hybrid rule allocation scheme in software defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9.
[12] Q. Li, N. Huang, D. Wang, X. Li, Y. Jiang, and Z. Song, "HQTimer: A hybrid *Q*-learning-based timeout mechanism in software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 1, pp. 153–166, Mar. 2019.
[13] J.-P. Sheu and Y.-C. Chuo, "Wildcard rules caching and cache replacement algorithms in software-defined networking," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 19–29, Mar. 2016.

[14] T. Cheng, K. Wang, L.-C. Wang, and C.-W. Lee, "An in-switch rule caching and replacement algorithm in software defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[15] S. H. Rastegar, A. Abbasfar, and V. Shah-Mansouri, "On fair rule caching in software defined radio access networks," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 460–463, Jun. 2018.
[16] J.-F. Huang, G.-Y. Chang, C.-F. Wang, and C.-H. Lin, "Heterogeneous flow table distribution in software-defined networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 252–261, Apr. 2016.
[17] I. I. Awan, N. Shah, M. Imran, M. Shoaib, and N. Saeed, "An improved mechanism for flow rule installation in in-band SDN," *J. Syst. Archit.*, vol. 96, pp. 1–19, Mar. 2019.
[18] Y. Liu, D. Niu, and B. Li, "Delay-optimized video traffic routing in software-defined interdatacenter networks," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 865–878, May 2016.
[19] J. Zheng, B. Li, C. Tian, K.-T. Foerster, S. Schmid, G. Chen, J. Wu, and R. Li, "Congestion-free rerouting of multiple flows in timed SDNs," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 968–981, May 2019.
[20] H. Xu, Z. Yu, X.-Y. Li, L. Huang, C. Qian, and T. Jung, "Joint route selection and update scheduling for low-latency update in SDNs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3073–3087, Oct. 2017.
[21] M. Paliwal and D. Shrimankar, "Effective resource management in SDN enabled data center network based on traffic demand," *IEEE Access*, vol. 7, pp. 69698–69706, 2019.
[22] R. Maaloul, R. Taktak, L. Chaari, and B. Cousin, "Energy-aware routing in carrier-grade Ethernet using SDN approach," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 3, pp. 844–858, Sep. 2018.
[23] R. Chai, H. Li, F. Meng, and Q. Chen, "Energy consumption optimization-based joint route selection and flow allocation algorithm for software-defined networking," *Sci. China Inf. Sci.*, vol. 60, no. 4, pp. 1–14, Mar. 2017.
[24] X. Yang, H. Xu, L. Huang, G. Zhao, P. Xi, and C. Qiao, "Joint virtual switch deployment and routing for load balancing in SDNs," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 397–410, Mar. 2018.
[25] Y.-C. Wang and S.-Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1422–1434, Dec. 2018.
[26] X. Xiaolong, C. Yun, H. Liuyun, and K. Anup, "MTSS: Multi-path traffic scheduling mechanism based on SDN," *J. Syst. Eng. Electron.*, vol. 30, no. 5, pp. 974–984, Oct. 2019.
[27] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 351–362, Aug. 2010.
[28] H. Huang, S. Guo, P. Li, B. Ye, and I. Stojmenovic, "Joint optimization of rule placement and traffic engineering for QoS provisioning in software defined network," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3488–3499, Dec. 2015.
[29] M. T. I. Ul Huque, G. Jourjon, C. Russell, and V. Gramoli, "Software defined networks garbage collection with clean-up packets," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1595–1608, Dec. 2019.
[30] S. Bera, S. Misra, and A. Jamalipour, "FlowStat: Adaptive flow-rule placement for per-flow statistics in SDN," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 530–539, Mar. 2019.
[31] P. Rezende, S. Kianpisheh, R. Glitho, and E. Madeira, "An SDN-based framework for routing multi-streams transport traffic over multi-path networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[32] L. Kleinrock, *Queueing Systems I: Theory*. New York, NY, USA: Wiley, 1975.
[33] D. E. Chua, E. D. Kolaczyk, and M. Crovella, "Efficient monitoring of end-to-end network properties," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, May 2005, pp. 1701–1711.
[34] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, Mar. 2017.
[35] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, "The traveling salesman problem: A guided tour of combinatorial optimization," *J. Oper. Res. Soc.*, vol. 37, no. 5, pp. 535–536, 1986.
[36] G. Scano, M.-J. Huguet, and S. U. Ngueveu, "Adaptations of K-shortest path algorithms for transportation networks," in *Proc. Int. Conf. Ind. Eng. Syst. Manage. (IESM)*, Oct. 2015, pp. 663–669.
[37] P. Bogdan, "Dijkstra algorithm in parallel-case study," in *Proc. 16th Int. Carpathian Control Conf. (ICCC)*, Jun. 2015, pp. 50–53.

LEI LUO received the B.E. degree from Panzhihua University, Sichuan, China, in 2018. He is currently pursuing the M.S. degree with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include resource virtualization, and resource management of wireless and mobile networks.

RONG CHAI (Senior Member, IEEE) received the B.E. and M.S. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering from McMaster University, ON, Canada, in 2008. In 2008, she joined the School of Communication and Information Engineering, Chongqing University of Posts and Technology, where she is currently a Professor. She has authored or coauthored 95 research articles. Her research interests include wireless communication and network theory.

QIONGFANG YUAN received the B.E. degree from Shaoyang University, Hunan, China, in 2017. She is currently pursuing the M.S. degree with the Chongqing University of Posts and Telecommunications, Chongqing, China. Her research interest includes resource management for software-defined networking.

JINYAN LI received the master's degree from the Beijing University of Posts and Telecommunications. She is currently a Senior Engineer with the China Telecom Technology Innovation Center. Her research interests include standardization and technique evolution of mobile networks.

CHENGLI MEI received the Ph.D. degree from Shanghai Jiaotong University. He is currently the Director and a Professorate Senior Engineer of the China Telecom Technology Innovation Center. His research interest includes techniques of mobile communication networks.