electronics

MDPI

Communication

Mobility-Aware Hybrid Flow Rule Cache Scheme in Software-Defined Access Networks

Youngjun Kim [1], Jinwoo Park [1] and Yeunwoong Kyung [2,*]

[1] Department of Electrical Engineering, Korea University, Seoul 02841, Korea; yeongjun2@korea.ac.kr (Y.K.); jwpark@korea.ac.kr (J.P.)
[2] School of Computer Engineering, Hanshin University, Osan 18101, Korea
* Correspondence: ywkyung@hs.ac.kr

Abstract: Due to the dynamic mobility feature, the proactive flow rule cache method has become one promising solution in software-defined networking (SDN)-based access networks to reduce the number of flow rule installation procedures between the forwarding nodes and SDN controller. However, since there is a flow rule cache limit for the forwarding node, an efficient flow rule cache strategy is required. To address this challenge, this paper proposes the mobility-aware hybrid flow rule cache scheme. Based on the comparison between the delay requirement of the incoming flow and the response delay of the controller, the proposed scheme decides to install the flow rule either proactively or reactively for the target candidate forwarding nodes. To find the optimal number of proactive flow rules considering the flow rule cache limits, an integer linear programming (ILP) problem is formulated and solved using the heuristic method. Extensive simulation results demonstrate that the proposed scheme outperforms the existing schemes in terms of the flow table utilization ratio, flow rule installation delay, and flow rules hit ratio under various settings.

Keywords: flow rule; hybrid rule cache; mobile flow; software-defined networking

Citation: Kim, Y.; Park, J.; Kyung, Y. Mobility-Aware Hybrid Flow Rule Cache Scheme in Software-Defined Access Networks. *Electronics* **2022**, *11*, 160. https://doi.org/10.3390/electronics11010160

Academic Editors: Danda B. Rawat and Cheng-Chi Lee

Received: 3 December 2021
Accepted: 3 January 2022
Published: 5 January 2022

1. Introduction

Software-defined networking (SDN) decouples the control plane from the forwarding plane via a programmable interface between two planes, such as OpenFlow [1]. The separated SDN architecture supports a global network view and flow-based fine-granular traffic control. Based on the separated architecture, SDN has been researched with respect to enhancement of network operation efficiency. For example, SDN enables unified control, which is supported by the device, edge, and orchestration controllers [2]. In addition, SDN allows heterogeneous layered networks in terms of radio nodes as well as operators to be optimized by the SDN orchestrator [3]. Moreover, in combination with mobile edge computing (MEC), SDN has been applied in 5G networks to design dynamic, manageable, and cost-effective networks. For instance, SDN can perform the decision making for MEC server selection to maximize the perceived users' satisfaction as well as the MEC servers' profit based on the global view of the SDN controller [4]. Furthermore, it is expected that the flexible operation of SDN can be extended to consider end-to-end network orchestration with heterogeneous components, domains, and relevant interfaces [5]. In this way, it can be noted that SDN can provide benefits in the field of communications and computing with respect to the perspectives of both users and network operators [4].

Basic data delivery in SDN is performed by matching the flow rules in the forwarding plane nodes stored by the controller. When there is no matched rule for the incoming flow, the forwarding node asks for rule caching to the controller using the packet-in message. Then, the controller caches the flow rule to the forwarding nodes along with the appropriate path using the packet-out or flow-mod messages. This procedure is the reactive flow rule placement operation [1].

In the forwarding plane, flow rules are cached in the local Ternary Content Addressable Memory (TCAM), which is expensive and has a limited size [6]. Therefore, the efficient rule caching in the TCAM for providing flow-based fine-granular traffic control has become one of the most important research issues in SDN.

Considering the rule placement problem with the capacity constraint of the forwarding node, conventional studies have provided rule compression [7], distribution [8], and energy-aware rule caching [9]. However, these studies focused on the static scenarios without considerations of the mobile environment [10].

On the other hand, SDN is attracting interest to support mobile scenarios such as software-defined Vehicular Access Networks (VANs) [11] and the Internet of Things [12]. In mobile scenarios, there can be dynamic flow rule changes due to the device's mobility features. Figure 1 shows the motivating example of the mobile scenario. Since it is a challenging task to implement virtualized wireless control functions [13], such as resource and security management (i.e., they can be implemented in the access devices (ADs) or in the controller decoupled from the ADs), considering the wireless parts of ADs will be one of our future works. Note that ADs communicate with the controller based on the SDN interface (i.e., OpenFlow [1]), receive packets from mobile nodes (MNs), and deliver them to the core network according to the flow rules. Initially, upon receiving the new packets of flow 1 and flow 2 from the MN, AD1 encapsulates the packets in a packet-in message and sends it to the controller using the OpenFlow protocol [1]. The controller then makes the rules for flow 1 and 2 to AD1. This means that two flow rules are installed by means of the reactive flow rule placement operation as described above. We assume that flow 1 is delay-sensitive and flow 2 was delay-tolerable. According to the two flow rules, the data of the two flows can be delivered through AD1. After the MN moves to AD2's area, the MN tries to start the communication with AD2 for flows 1 and 2. In this case, if the flow rules for flows 1 and 2 are again installed by means of the reactive operation, the response delay for installation is required. In addition, if there are other flow rule installation requests at the same time, the delay becomes higher, which can result in quality of service (QoS) degradation, especially for the delay-sensitive flow (i.e., flow 1). Although the mobility management protocols have been applied in the SDN-based access networks, such as the distributed mobility management [14] and proxy mobile IPv6 [15], to preserve the IP address during mobility, the response delay issue is unavoidable because such protocols are operated in the control plane after the forwarding plane delivers the packet-in messages to the controller [16,17]. To handle the response delay issue, several works on proactive flow rule caching considering the mobility features have been conducted [6,11,12,18–24]. These works proactively cache the flow rules to the predicted target forwarding nodes, utilizing mobility prediction models such as the Markov predictor [12,18,19,21] and machine learning [20]. Since these models naturally have a prediction error according to variable factors, some of the proactively cached rules can be a waste of resources, being repetitively cached with multiple forwarding nodes, and are not utilized at all. In addition, these works focused on the proactive rule cache method based on mobility prediction without considerations of the load status of the controller. This means that they proactively cache the flow rules even when the fast response delay of the controller is guaranteed. In other words, some proactively cached flow rules in these works can be uncertain, even though the controller can manage the rules with certainty based on the intelligence and network view. As mentioned above, because of the location prediction error, reactive flow rule placement can be more efficient than the proactive approach if the controller's response delay meets the delay requirements of the flows.

To address the problems above, this paper proposes a hybrid flow rule cache scheme which proactively as well as reactively caches the flow rules based on the response delay of the controller and delay requirements of the flows. After we formulate the proposed hybrid flow rule cache problem to minimize the waste of resources, extensive simulation results are provided which show that the proposed scheme can reduce the waste of resources and enhance the flow rule hit ratio compared with the previous works.

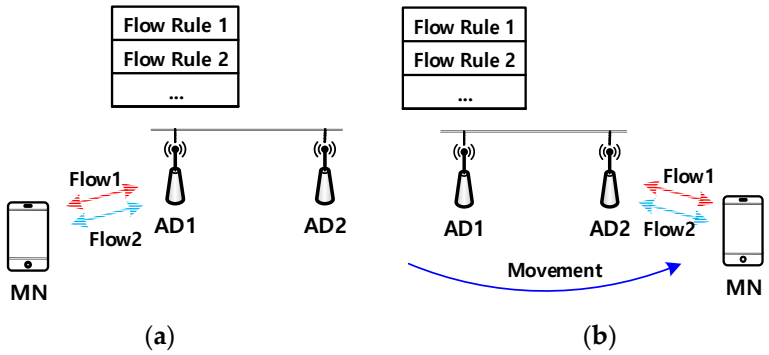

Figure 1. Motivating example. (**a**) Two flows of the MN are delivered through AD1. (**b**) MN moves and starts to communicate with AD2.

The key contributions of this paper are twofold: (1) to the best our knowledge, this is the first work to optimize the performance of the flow rule cache problem with the consideration of the response delay from the controller, which can be a key performance metric to determine whether the proactive flow rule cache is required or not, and (2) we evaluate the proposed hybrid flow rule cache scheme in terms of the flow table utilization ratio, flow rule installation delay, and flow rules hit ratio under various environments, which can provide valuable design guidelines for advanced software-defined access networks.

The remainder of this paper is organized as follows. The related works on the rule placement problem are presented in Section 2. Then, the system model and proposed hybrid cache scheme are described in Sections 3 and 4, respectively. After Section 5 presents the performance evaluation, Section 6 concludes this paper.

2. Related Work

The rule placement problem was researched in the SDN field for two main reasons: capacity constraint and signaling overhead. For flow-based fine-granular traffic engineering, numerous flow rules are required. For example, a typical enterprise network requires up to 8 million rules [25]. As mentioned above, flow rules are usually cached in the TCAM on the forwarding nodes, which enables very high lookup performance. However, the TCAM is 400 times more expensive than traditional memory and has limited capacity [26]. This results in approximately 20,000 flow rules for the commercial off-the-shelf switches, which is much less than the number of required rules for efficient traffic engineering [27]. In addition, as explained above, the flow rule caching requires the signaling exchange between the forwarding nodes and SDN controller. Consequently, the SDN controller can be overloaded when there are excessive flow rule request messages. This situation becomes severe in the access networks due to the dynamic mobility feature, which results in numerous mobile flows (i.e., handover events).

To handle the rule placement problem, there have been efforts to reduce the number of flow rules in the core network devices. B. Wolfgang et al. [7] utilized the compression of the flow rules by means of wildcard expressions and logic minimization to save the flow entries without much compression time. X-N. Nguyen et al. [8] provided the linear programming model for the flow rule allocation problem to satisfy the endpoint network policy while relaxing the routing policy. F. Giroire et al. [9] presented an optimization scheme to minimize the energy consumption of the routers and guarantee the QoS requirements based on the centralized network view. L. Luo et al. [10] formulated an optimization problem for the rule caching method to minimize the end-to-end delay and provided a heuristic algorithm to solve it. However, these works only considered the static scenarios in the core network. In other words, they only assumed that the new flows which required the flow rule installation were newly generated flows to the devices without consideration for the moved flows from other devices due to the handover. Since the requirements between the newly generated and moved flows can be different in the access networks [11,12],

an efficient method for flow rule installation that considers the different requirements is required. Note that this paper also covers the differentiation between the newly generated and moved flows.

Compared with these works, which focused on the static scenarios in the core networks, there have been several works which considered mobile scenarios in access networks [6,11,12,18–24]. Since there can be numerous mobile flows (i.e., handover events), they may generate multitudinous flow rule requests and necessitate a fast response for flow rule installation. To meet these challenges, mobility has been considered a main feature of the flow rule placement. As an initial work, H. Li et al. [6] used duplicated rules according to the user mobility while minimizing the rules' space occupation and guaranteeing the satisfaction ratio. However, this work just focused on the constraint of flow rules without consideration of the flow rule installation procedure between the devices and controller. S. Misra et al. [11] proactively cached the flow rules into a new path which would be utilized for the computed task download according to the user's mobility. S. Bera et al. [12] predicted the next locations of the end users based on the Markov predictor and determined the access points (APs) to minimize the delay and the number of active APs. Similarly, M. Dong et al. [18] provided prefetching algorithms which optimized the least recently used approach, considering the forwarding paths in the access networks and user positions to increase the cache hit ratio. X. Wang et al. [19] utilized multicast addresses and installed the rules in advance based on the conditions of mobile users to decrease the number of rules for the Internet of Vehicle (IoV) scenarios. Although these papers [11,12,18,19] optimized the number of flow rules in the devices, they did not consider the controller's response delay. This means that the flow rules were proactively installed even when the controller could guarantee the requirements of flow rule installation. This can result in inefficiency of the resource utilization. E. Zelikovic et al. [20] also predicted the future location and AP load using a recurrent neural network (RNN) and then migrated the virtual AP to achieve seamless handover. L. Mendiboure et al. [21] introduced a flow rule deployment policy according to the flow table occupancy rate, mobility of vehicles, and control channel capacity. In addition, an optimal selection for the flow rule path in the fog computing of the offloading scenario was considered to reduce the average delay and energy consumption based on the flow rule capacity constraint [22]. Moreover, T. Theodorou et al. [23] invoked global topology discovery processes to detect the mobility and updated flow rules proactively based on the centralized global network view. S. H. Rastegar et al. [24] configured an integer linear program to handle the flow table resource allocation problem when the traffic was generated in a bursty on-off manner. However, these papers [20–24] utilized the proactive rule placement method only. Consequently, if a prediction error occurred which was a natural problem of the prediction algorithms [18], high resource usage (i.e., repetitively cached at multiple forwarding devices) could be experienced, which could block the additional flow rule installations.

To handle the above problems, this paper considers the response delay of the controller, which can be a key performance metric to determine whether a proactive flow rule cache is required or not. In addition, since the prediction error of the proactive flow rule cache method can lead to a waste of resources, the reactive flow rule cache method can be efficient if the controller's response delay meets the delay requirements of the flows. Therefore, this paper proposes a hybrid flow rule cache scheme which proactively as well as reactively caches the flow rules based on the response delay of the controller and the delay requirements of the flows. In the proposed scheme, the reactive flow rule cache method is preferred to prevent the waste of flow rule caches when the delay requirement of flows can be guaranteed by the response delay of the controller. On the other hand, the proactive flow rule cache method is utilized to remove the flow rule installation procedure, which can be a time-consuming task when the controller is overloaded.

3. System Model

Figure 2 presents the system architecture of software-defined access networks, where the forwarding nodes, including the ADs and switches, are connected to the controller via the SDN interface (i.e., OpenFlow [1]). Since the controller placement problem is one of the most challenging issues in SDN [28,29], this paper assumes that the controller can communicate with the ADs with a constant propagation delay through the dedicated control channel [30]. In addition, the controller can configure the flow-based network context view using the statistics reports from the forwarding nodes in terms of the port, queue, group, meter, and table [1,28]. Each forwarding node has its own flow table composed of flow rules. Mobile nodes such as sensor nodes (SNs), phones, and vehicles can move between the ADs.

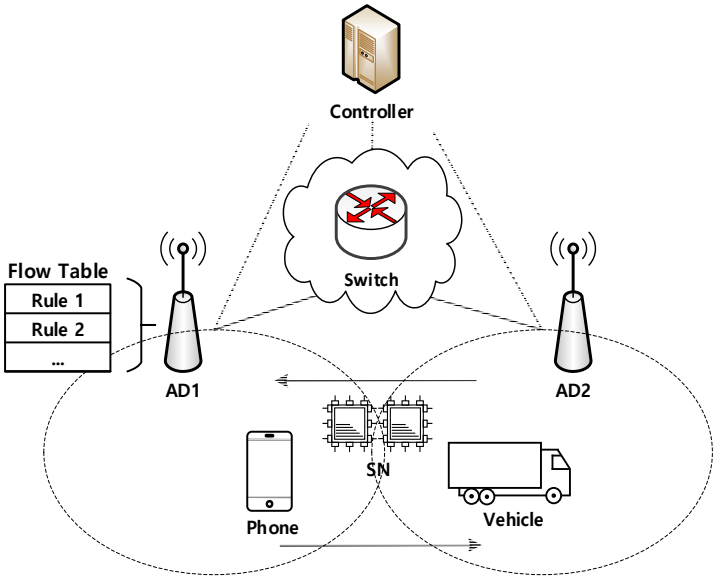

Figure 2. System architecture of the software-defined access network with mobile nodes.

When the mobile nodes move from AD1 to AD2, the flow rules of the mobile nodes can be proactively or reactively cached to AD2. If the proactive rule placement operates, the rules are proactively cached to AD2 and switch along with the path, which will be the changed path due to the mobility.

The number of flow rules at time t in the ith AD is denoted by R_i^t. The maximum number of rules in the ith access device due to the capacity constraint is denoted by $R_i{}^{max}$. R_i^t includes the reactively and proactively cached rules and is thus given by

$$R_i^t = R_i^{t-1} + R_{i,N}^t + R_{i,M}^t - R_{i,E}^t. \quad (1)$$

where $R_{i,N}^t$, $R_{i,M}^t$, and $R_{i,E}^t$ are the numbers of static, mobile, and expired flow rules, respectively, in the ith AD. Static flow rules are newly generated flows in the area of the AD. In addition, flow rules can be removed after a timeout with no received traffic according to the flow expiry mechanism in the OpenFlow protocol [1]. In addition, the mobile flows are handover flows from the neighboring ADs. The controller determines that the rules for the mobile flows need to be proactively or reactively cached. Therefore, $R_{i,M}^t$ is denoted by

$$R_{i,M}^t = \sum_{j=1}^{M_i^t} r_i^{r,j} + \sum_{j=1}^{M_i^t} r_i^{p,j}. \quad (2)$$

where $r_i^{r,j}$ and $r_i^{p,j}$ denote the jth reactive and proactive flow rules among the total M_i^t mobile flows into the ith AD at time t, while $r_i^{r,j}$ and $r_i^{p,j}$ are binary variables, where the

value is 1 if the flow rule exists and 0 if not. For example, if the kth flow rule is proactively installed in the ith AD, then $r_i^{r,k}$ is 0 and $r_i^{p,k}$ is 1.

Meanwhile, the jth mobile flow can have its own response delay requirement (or constraint) d_j^{th}. The response delay of the jth mobile flow can be measured by the duration between the packet-in and packet-out messages at t, denoted as d_j^t. This depends on the load on the controller. Since there can be a dynamic incoming load on the controller, especially for the mobile access networks, the delay requirement should be managed well. This is tightly related to the quality of service (QoS) [31]. In other words, a proactive rule cache is required when the controller cannot guarantee the delay requirement. If the flow rule is proactively cached, d_j^t can be 0, which means no delay. In addition, the controller's response delay at t with the ith AD is denoted as D_i^t from the controller's perspective.

The response delay depends on several parameters, such as a the queuing delay, transmission delay, and propagation delay. It can be noticed that the controller can monitor the queuing delays according to the current state. In this paper, we assume that the controller follows the M/M/1 queuing model, as it is generally utilized for SDN controllers [10]. Then, the average response delay, except for the transmission and propagation delays, is given by $1/(C - \lambda)$ according to Little's law [32], where C is the processing capacity of the controller and λ is the packet-in message arrival rates.

In addition, the transmission delay can be calculated using the bandwidth of the control channel and the sizes of packet-in and packet-out (or flow-mod) messages. Moreover, the propagation delay can be affected by the medium type. These parameters can be checked using the network statistics reports from the forwarding nodes defined by the OpenFlow protocol [1]. Therefore, the controller can estimate the current response delay using the queuing, transmission, and propagation delays [33] and monitor the history of the response delays. However, we need to know the future response delay to determine whether to proactively cache rules or not. Therefore, based on the history of the response delays, which can be calculated as described above, estimation of the future delay can be conducted as follows. We assume that there are discrete delay levels which are matched one to one with the calculated delays for the scalable operation. This paper utilizes a long short-term memory (LSTM) neural network to forecast the delay, which is widely utilized for load and state prediction [20,34,35]. The core component of LSTM is to use memory cells which are composed of three controlling gates (Figure 3): the input gate (i_t), forget gate (f_t), and output gate (o_t) at time t. The input gate decides to add the input data to the memory cell. In addition, the forget gate aims to forget or reset the last state of the memory cell. Moreover, the output gate controls the output data that can be propagated to the rest of the network. The equations for forward propagation are given by

$$\begin{pmatrix} f_t \\ i_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tan h \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{4}$$

$$h_t = o_t \odot \tan h(c_t) \tag{5}$$

where x_t, c_t, and h_t are the input sequence, memory cell state, and hidden state at t, respectively. In addition, σ, W, and $\odot$ denote the sigmoid function which maps the input data to the range between 0 and 1, weight matrix, and the Hadamard product, respectively [34]. The input data sequence for the LSTM network are the response delays from the controller, which are determined by the packet-in messages at each time unit. Then, the output is the predicted response delay. An Adam optimizer is used to stochastically optimize the parameters in the model.

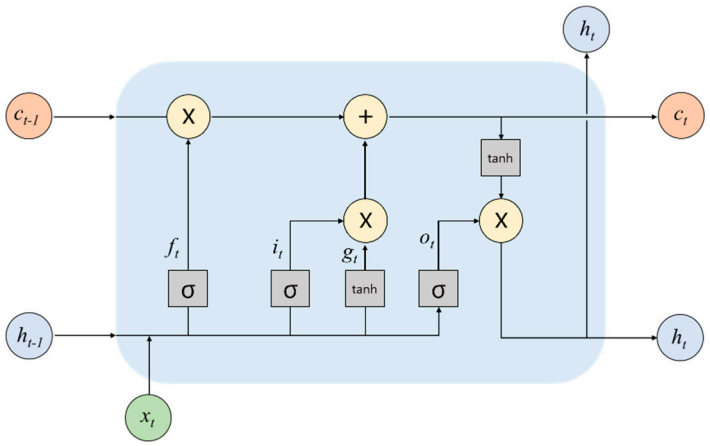

Figure 3. The structure of the LSTM network.

4. Proposed Hybrid Flow Rule Cache Scheme

In this section, the proposed hybrid flow rule cache scheme is described. Let us consider that there are N number of ADs and M_i number of mobile flows into the ith AD. This paper assumes that the predicted target ADs are determined for the mobile flows. To find the optimal number of proactive flow rules, the proposed scheme can be formulated as an integer linear programming (ILP) problem, and the problem can be solved using the heuristic method.

4.1. Problem Formulation

Where $r_i^{p,j}$ denotes the jth proactive flow rule among the total M_i^t mobile flows into the ith AD at time t, the ILP problem for the hybrid flow rule cache scheme is formulated as follows:

$$\text{Minimize } \sum_{i=1}^{N} \sum_{j=1}^{M_i^t} r_i^{p,j} \tag{6}$$

$$\text{subject to } R_i^t \leq R_i^{max} \tag{7}$$

$$d_j^t \leq d_j^{th} \tag{8}$$

$$r_i^{p,j} = 1 \; if D_i^t \geq d_j^{th} \tag{9}$$

$$r_i^{p,j} = 0 \; if D_i^t < d_j^{th} \tag{10}$$

Since the reactive flow rules should be installed because they are from actual moved flows after handover into each AD, the proposed scheme aims to minimize the proactive flow rules (i.e., predicted flow rules) as shown in Equation (6). Equation (7) denotes that the total number of flow rules at time t in the ith AD is always less than or equal to its flow rule capacity constraint. In Equation (7), the total number of flow rules includes static, mobile (reactively and proactively added rules), and expired flow rules as described in Equation (1). Equation (8) represents that the response delay from the flow's perspective should be less than or equal to its response delay constraint, and d_j^t can be expressed according to the existence of the proactively cached rule as follows:

$$d_j^t = \begin{cases} 0 \; if \; r_i^{p,j} = 1 \\ D_i^t \; if \; r_i^{p,j} = 0 \end{cases}. \tag{11}$$

Equations (9) and (10) denote that the controller proactively caches the rule by comparing its delay performance with the flow's delay constraint.

Since the above problem needs delay estimation for each flow, the time complexity of the proposed scheme is $O(N \times w \times M^2)$, where w is the number of weights.

4.2. Heuristic Algorithm

Since the above ILP problem is NP hard [36], this paper provides a simple heuristic algorithm (Algorithm 1) for the hybrid flow rule cache scheme. It is assumed that the proactive rule caching is determined before the mobile flow physically moves to the target AD and multiple target ADs can be predicted based on the prediction methods. As we explained above, since the reactive flow rules should be installed in an AD as they are from actual moved flows after handover into the area of each AD, the algorithm aims to determine whether the proactive flow rule for mobile flows is needed or not based on the estimated controller's response delay and delay requirement of flows.

Algorithm 1 Heuristic hybrid flow rule algorithm

Input: Set of ADs N, Set of mobile flows M_i from ith AD, Maximum rule capacity R_i^{max} of ith AD, Delay constraint of jth flow d_j^{th}.
Output: Flow tables including proactive flow rules in ADs
1: for $i \in N$ do
2: Estimate the controller's response delay D_i
3: for $j \in M_i$ do
4: Select the flow j which has lowest d_j^{th}
5: if $D_i > d_j^{th}$ then
6: if $R_i < R_i^{max}$ then
7: Insert flow j rule in ith AD
8: $R_i = R_i + 1$;

For each AD, the controller first estimates the response delay (i.e., not for each flow), which can reduce the complexity. Then, among the flows in each AD, the controller finds the flow which has the minimum delay constraint, because a low delay requirement means that the flow is delay-sensitive. After the delay constraint of the flow is compared with the estimated controller's response delay, whether proactive rule caching is performed or not is determined. After the decisions of the mobile flows are completed, the static flows are then processed. Although the delay constraint for the static flows can also be considered, the controller handles them without priority as much as possible for simplicity. The time complexity of the proposed heuristic hybrid flow rule algorithm can be expressed as $O(N \times w \times M \times \log M)$.

4.3. Implementation Cost and Computational Complexity

This paper considers that the implementation cost of each decision (i.e., whether to use the reactive or proactive flow rule cache) includes the processing cost, prediction cost, and transmission cost [37]. In the proposed scheme, the processing cost is dependent on the number of mobile flows multiplied by the number of ADs (i.e., $M \times N$). After the prediction based on LSTM, whose cost is dependent on the number of weights (i.e., w) [38], flow rules can be installed one by one in ascending order by delay constraint (i.e., $\log M$) in the heuristic algorithm. On the other hand, they can be installed by comparing every flow with each other (i.e., M) in the ILP problem. Consequently, the computational complexity of the proposed scheme for the ILP problem and heuristic algorithm can be expressed as $O(N \times w \times M \times M)$ and $O(N \times w \times M \times \log M)$, respectively.

5. Performance Evaluation

We evaluated the performance of the proposed hybrid flow rule cache scheme compared with the basic reactive flow rule cache method [1] and MobiFlow [12]. The simulation parameters are presented in Table 1. We considered the Abilene WAN topology in Internet2 [39], including 12 switches which were connected to the SDN controller. We assumed that each switch had 20,000 flow rules [27]. This paper used the M/M/1 queuing model for the SDN controller as explained in Section 3, where the processing capacity of the controller was set to 5000. In addition, the propagation delay from the forwarding nodes to the

controller through the dedicated control channel was set to 1 ms. Therefore, the response delay of the controller depended on the queuing delay. In addition, this paper adapted Poisson–Voronoi tessellation with the random way point mobility model [40]. According to the mobility model, there were 10 mobile users whose average packet-in message arrival rate was 300 based on the Poisson process. Evaluation results were drawn by generating 50,000 random numbers of the packet-in message arrival rate (with an average of 300), location, velocity (between 0 m/s to 50 m/s), and direction (between 0 to 360 degrees) in the Abilene WAN topology. In addition, the average delay constraint for the delay-sensitive flows was assumed to be 30 ms [18]. Moreover, to train the LSTM network, randomly generated packet-in messages according to the distribution at each time unit were utilized to calculate the response delay as the input data. Then, the LSTM network could predict the next response delay. This paper used 75% of the input data to train the network and 25% to test the network. For the training, the number of features, gradient threshold, and number of epochs were 1, and 30, respectively.

Table 1. Simulation parameters.

Parameters	Value
Processing capacity of the controller (C)	5000 messages/second
Flow rule capacity	20,000
Network topology	Abilene WAN [39]
Mobility model	Poisson–Voronoi tessellation [40]
The number of MNs	10
Packet-in message arrival rate per MN (λ)	300
Bandwidth of the control channel	100 Mbps
Propagation delay	1 ms
Mobility prediction error rate	20%
Ratio of delay sensitive flows	50%
Delay constraint of delay sensitive flows	30 ms

Figure 4 shows the average flow table utilization ratio per AD according to the flow arrival rate of each mobile user. This is the average ratio of flow rule usage to the max capacity of flow entries. The reactive flow rule cache scheme had minimal flow rules because it did not include the proactively cached rules. Since the proposed scheme proactively cached rules only for flows which had a lower delay constraint than the controller's response delay, it could reduce the proactively cached rules compared with MobiFlow. For example, if the flow arrival rate was 300, the proposed scheme could save approximately 11 percent of the flow table space compared with MobiFlow. As the number of flows increased, the gap between the proposed scheme and MobiFlow decreased, because the controller's response delay became higher according to the number of flows. This means that the number of flows whose delay constraints were not satisfied also increased.

Table 2 shows the average response delay according to the flow arrival rate. It can be noticed that the optimal solution of the proposed scheme (i.e., Pro(O)), which was derived through the above ILP problem, had lower values than those of the heuristic solution of the proposed scheme (i.e., Pro(H)), because the optimal solution estimated the response delay precisely per flow. When the flow arrival rate increased, the average response delay also increased for all schemes because of the maximum capacity constraint of the flow entries. However, the proposed scheme had lower values than MobiFlow, because the proposed scheme could retain more available space for the flow rules as described in Figure 4. In the case of the reactive flow cache scheme, since it always required the controller's interaction, it had the highest delay compared with the other schemes. Based on Figure 4 and Table 2, it can be noted that the proposed scheme could support the delay requirement of the flows (i.e., by using the proactive operation) while maintaining more available resources in the flow table (i.e., due to the reactive operation), which could be interpreted as the efficiency of the hybrid scheme.

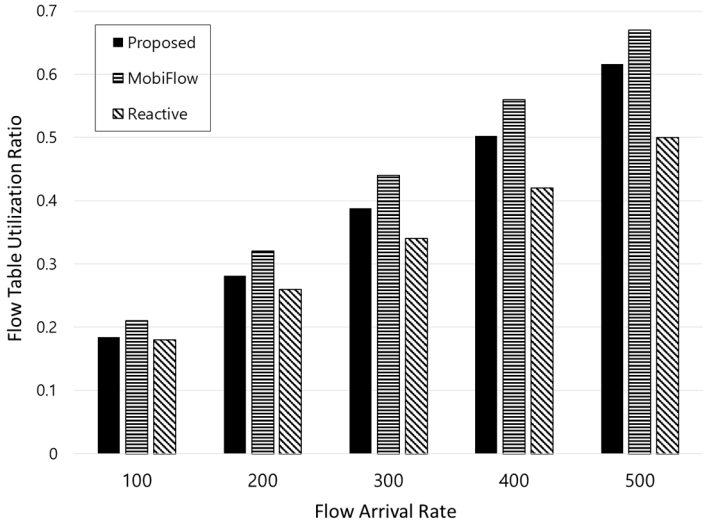

Figure 4. Flow table utilization ratio according to the flow arrival rate of each mobile user.

Table 2. Average response delay (ms) according to the flow arrival rate.

λ	Pro(O)	Pro(H)	MobiFlow	Reactive
300	29.65	30.82	31.21	41.52
400	30.06	32.39	33.21	46.91
500	33.68	36.77	39.79	57.97

Table 3 shows the time complexity according to the flow arrival rate. Since the computation complexity of the ILP problem (i.e., Pro(O)) depended on M^2, the running time could increase quadratically according to the number of MNs. Consequently, due to its high computational complexity, the optimal solution for the ILP problem could be determined only with a small number of MNs [41]. On the other hand, because the computational complexity of the heuristic algorithm (i.e., Pro(H)) depended on $M\log M$, it could be practically considered in the real SDN environment [42].

Table 3. Computational complexity according to the flow arrival rate.

λ	Pro(O)	Pro(H)
300	O(30,000)	O(3000log10)
400	O(40,000)	O(4000log10)
500	O(50,000)	O(5000log10)

Figure 5 displays the flow rules hit ratio according to the mobility prediction error per AD, which is the ratio of the hit flow rules to the total requested flows. The delay constraints for the proposed scheme were set to 50 ms and 30 ms (i.e., proposed (50 ms) and proposed (30 ms)). The flow rules hit ratio of the proposed scheme and MobiFlow became reduced according to the prediction error, because these schemes performed prediction-based proactive flow rule caches. Even though the prediction error existed, the proposed scheme had a higher hit ratio compared with MobiFlow because it utilized the proactive rule cache only when the response delay was not satisfied with the delay constraint. In addition, the flow rule hit ratio increased when the delay constraint of the flows increased. This was because the number of flows that did not need to be proactively cached increased.

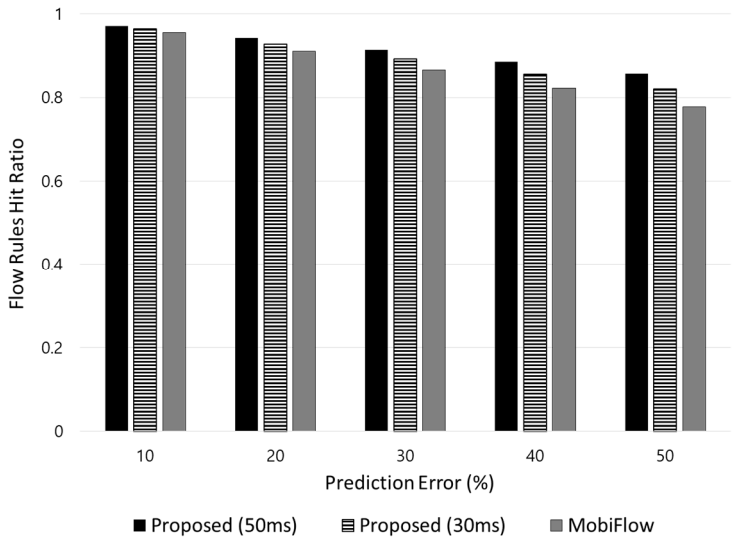

Figure 5. Flow rules hit ratio according to the mobility prediction error.

Note that the proposed scheme could have a lower risk of prediction error compared with MobiFlow because the proposed scheme utilizes the reactive flow rule cache method as much as possible based on the controller's response delay. However, if the prediction error became higher, the proposed scheme could also have a waste of resources due to the wrong prediction compared with the reactive flow rule cache method. Therefore, the operator should carefully determine whether to use the proactive rule cache method or not based on the prediction error, considering the trade-off between the perspectives of the resource efficiency and delay requirement support. Analysis of the trade-off according to the prediction error will be one of our future works.

6. Conclusions

In SDN-based access networks, the proactive rule cache method that does not need the flow rule installation procedure has been considered for the delay requirement of the mobile flows. However, due to the location prediction error which can result in the waste of flow rule caches, an efficient flow rule cache strategy is required. To address this problem, this paper proposed a mobility-aware hybrid flow rule cache scheme considering the delay requirements of flows and the response delay of the controller with the flow table capacity constraint. Based on the hybrid usage of reactive and proactive cache methods, the proposed scheme can efficiently utilize the flow rule caches while satisfying the delay requirement. The proposed scheme was formulated with ILP and solved with the heuristic algorithm. The simulation results demonstrate that the proposed scheme outperformed the existing schemes in terms of the flow table utilization ratio, average response delay, and flow rules hit ratio considering the flow arrival rate and prediction errors. For future works, the proposed scheme will be applied to the OpenFlow-based physical SDN platform to show the practical performance.

Author Contributions: Conceptualization, Y.K. (Youngjun Kim), J.P. and Y.K. (Yeunwoong Kyung); methodology, Y.K. (Yeunwoong Kyung); software, Y.K. (Youngjun Kim); validation, Y.K. (Youngjun Kim), J.P. and Y.K. (Yeunwoong Kyung); formal analysis, investigation, Y.K. (Yeunwoong Kyung); resources, data curation, J.P.; writing—original draft preparation, Y.K. (Yeunwoong Kyung); writing—review and editing, Y.K. (Youngjun Kim); visualization, Y.K. (Youngjun Kim); supervision, Y.K. (Yeunwoong Kyung); project administration, J.P.; funding acquisition, Y.K. (Yeunwoong Kyung). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No.2020R1G1A1100493).

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

References

1. OpenFlow Switch Specification 1.5.1. Available online: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed on 26 September 2021).
2. Guerzoni, R.; Trivisonno, R.; Soldani, D. SDN-based Architecture and Procedures for 5G Networks. In Proceedings of the 1st International Conference on 5G for Ubiquitous Connectivity, Akaslompolo, Finland, 26−28 November 2014.
3. Wang, M.; Karakoc, N.; Ferrari, L.; Shantharama, P.; Thyagaturu, A.S.; Reisslein, M.; Scaglione, A. A Multi-Layer Multi-Timescale Network Utility Maximization Framework for the SDN-Based LayBack Architecture Enabling Wireless Backhaul Resource Sharing. *Electronic* **2019**, *8*, 937. [CrossRef]
4. Mitsis, G.; Apostolopoulos, P.A.; Tsiropoulou, E.E.; Papavassiliou, S. Intelligent Dynamic Data Offloading in a Competitive Mobile Edge Computing Market. *Future Internet* **2019**, *11*, 118. [CrossRef]
5. Trakadas, P.; Sarakis, L.; Giannopoulos, A.; Spantideas, S.; Capsalis, N.; Gkonis, P.; Karkazis, P.; Rigazzi, G.; Antonopoulos, A.; Cambeiro, M.A.; et al. A Cost-Efficient 5G Non-Public Network Architectural Approach: Key Concepts and Enablers, Building Blocks and Potential Use Cases. *Sensors* **2021**, *21*, 5578. [CrossRef] [PubMed]
6. Li, H.; Li, P.; Guo, S. MoRule: Optimized Rule Placement for Mobile Users in SDN-enabled Access Networks. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014.
7. Wolfgang, B.; Menth, M.; Felter, W.; Dixon, C.; Carter, J. Wildcard compression of inter-domain routing tables for OpenFlow-based Software-Defined Networking. In Proceedings of the Third European Workshop on Software Defined Networks, Budapest, Hungary, 1−3 September 2014.
8. Nguyen, X.-N.; Saucez, D.; Barakat, C.; Turletti, T. OFFICER: A general Optimization Framework for OpenFlow Rule Allocation and Endpoint Policy Enforcement. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015.
9. Giroire, F.; Moulierac, J.; Phan, T.K. Optimizing Rule Placement in Software-Defined Networks for Energy-aware Routing. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014.
10. Luo, L.; Chai, R.; Yuan, Q.; Li, J.; Mei, C. End-to-End Delay Minimization-Based Joint Rule Caching and Flow Forwarding Algorithm for SDN. *IEEE Access* **2020**, *8*, 145227–145241. [CrossRef]
11. Misra, S.; Bera, S. Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network. *IEEE Trans. Veh. Netw.* **2020**, *69*, 2071–2078. [CrossRef]
12. Bera, S.; Misra, S.; Obaidat, M.S. Mobi-Flow: Mobility-Aware Adaptation Flow-Rule Placement in Software-Defined Access Network. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1831–1842. [CrossRef]
13. Kyung, Y.; Nguyen, T.M.; Hong, K.; Park, J.; Park, J. Software defined Service Migration through Legacy Service Integration into 4G Networks and Future Evolutions. *IEEE Commun. Mag.* **2015**, *53*, 108–114. [CrossRef]
14. Raza, S.M.; Kim, D.S.; Shin, D.; Choo, H. Leveraging proxy mobile ipv6 with SDN. *J. Commun. Netw.* **2016**, *18*, 460–475. [CrossRef]
15. Nguyen, T.T.; Bonnet, C.; Harri, J. SDN-Based Distributed Mobility Management for 5G Networks. In Proceedings of the IEEE Wireless Communications and Networking Conference, Doha, Qatar, 3–6 April 2016.
16. Bi, Y.; Han, G.; Lin, C.; Guizani, M.; Wang, X. Mobility Management for Intro/Inter Domain Handover in Software-Defined Networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1739–1754. [CrossRef]
17. Garg, S.; Kaur, K.; Ahmed, S.H.; Bradai, A.; Kaddoum, G.; Atiquzzaman, M. MobQoS: Mobility-Aware and QoS-Driven SDN Framework for Autonomous Vehicles. *IEEE Wirel. Commun.* **2019**, *26*, 12–20. [CrossRef]
18. Dong, M.; Li, H.; Ota, K.; Xiao, J. Rule Caching in SDN-enabled mobile access networks. *IEEE Netw.* **2015**, *29*, 40–45. [CrossRef]
19. Wang, X.; Wang, C.; Zhang, J.; Zhour, M.; Jiang, C. Improved Rule Installation for Real-Time Query Service in Software-Defined Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 225–235. [CrossRef]
20. Zelikovic, E.; Krijestorac, N.S.; Latre, S.; Barja, J.M.M. ABRAHAM: Machine Learning Backed Proactive Handover Algorithm Using SDN. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1522–1536. [CrossRef]
21. Mendiboure, L.; Chalouf, M.A.; Krief, F. Load-aware and Mobility-aware Flow Rules Management in Software Defined Vehicular Access Networks. *IEEE Access* **2020**, *8*, 167411–167424. [CrossRef]
22. Maity, I.; Misra, S.; Mandal, C. CORE: Prediction-Based Control Plane Load Reduction in Software-Defined IoT Networks. *IEEE Trans. Commun.* **2021**, *69*, 1835–1844. [CrossRef]
23. Theodorou, T.; Mamatas, L. SD-MIoT: A Software-Defined Networking Solution for Mobile Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 4604–4617. [CrossRef]
24. Rastegar, S.H.; Abbasfar, A.; S-Mansouri, V. Rule Caching in SDN-Enabled Base Stations Supporting Massive IoT Devices with Bursty Traffic. *IEEE Internet Things J.* **2020**, *7*, 8917–8931. [CrossRef]
25. Nguyen, X.-N.; Saucez, D.; Barakat, C.; Turletti, T. Rules Placement Problem in OpenFlow Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1273–1286. [CrossRef]

26. Katta, N.; Alipourfard, O.; Rexford, J.; Walker, D. Infinite CacheFlow in Software-defined Networks. In Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, New York, NY, USA, 22 August 2014.
27. Stephens, B.; Cox, A.; Felter, W.; Dixon, C.; Carter, J. PAST: Scalable Ethernet for Data Centers. In Proceedings of the ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT), New York, NY, USA, 10−13 December 2012.
28. Das, T.; Sridharan, V.; Gurusamy, M. A Survey on Controller Placement in SDN. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 472–503. [CrossRef]
29. Wang, G.; Zhao, Y.; Huang, J.; Wu, Y. An Effective Approach to Controller Placement in Software Defined Wide Area Networks. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 344–355. [CrossRef]
30. Savas, S.S.; Tornatore, M.; Dikbiyik, F.; Yayimli, A.; Martel, C.U.; Mukherjee, B. RASCAR: Recovery-Aware Switch-Controller Assignment and Routing in SDN. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 1222–1234. [CrossRef]
31. Kyung, Y.; Park, J. Prioritized admission control with load distribution over multiple controllers for scalable SDN-based mobile network. *Wirel. Netw.* **2019**, *25*, 2963–2976. [CrossRef]
32. Kleinrock, L. *Queueing Systems: Theory*; Wiley: New York, NY, USA, 1975; Volume 1.
33. Suh, D.; Pack, S. Low-Complexity Master Controller Assignment in Distributed SDN Controller Environments. *IEEE Commun. Lett.* **2017**, *22*, 490–493. [CrossRef]
34. Zhang, Y.; Song, X. Load Prediction of Space Deployable Structure Based on FBG and LSTM. *IEEE Access* **2019**, *7*, 13715–13722. [CrossRef]
35. Wang, W.; Zhou, C.; He, H.; Wu, W.; Zhuang, W.; Shen, X. Cellular Traffic Load Prediction with LSTM and Gaussian Process Regression. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020.
36. Pochet, Y.; Wolsey, L.A. *Production Planning by Mixed Integer Programming*; Springer: Berlin/Heidelberg, Germany, 2006.
37. Xu, Z.; Gong, W.; Xia, Q.; Liang, W.; Rana, O.F.; Wu, G. NFV-Enabled IoT Service Provisioning in Mobile Edge Clouds. *IEEE Trans. Mob. Comput.* **2021**, *20*, 1892–1906. [CrossRef]
38. Tsironi, E.; Barros, P.; Weber, C.; Wermter, S. An Analysis of Convolutional Long Short-Term Memory Recurrent Neural Networks for Gesture Recognition. *Neurocomputing* **2017**, *268*, 76–86. [CrossRef]
39. SNDlib. Available online: http://sndlib.zib.de/home.action (accessed on 26 September 2021).
40. Lin, X.; Ganti, R.K.; Fleming, P.J.; Andrews, J.G. Towards Understanding the Fundamentals of Mobility in Cellular Networks. *IEEE Trans. Wirel. Commun.* **2013**, *12*, 1686–1698. [CrossRef]
41. Liu, J.; Shi, Y.; Zhao, L.; Cao, Y.; Sun, W.; Kato, N. Joint Placement of Controllers and Gateways in SDN-Enabled 5G-Satellite Integrated Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 221–232. [CrossRef]
42. Bastam, M.; Sabaei, M.; Yousefpour, R. A scalable traffic engineering technique in an SDN-based data center network. *Trans. Emerg. Telecommun. Technol.* **2017**, *16*, e3268. [CrossRef]