Rule caching in SDN-enabled mobile access networks

SDN Rule Caching in Mobile Access Networks

Mianxiong Dong, *Member, IEEE,* He Li, *Member, IEEE,* Kaoru Ota, *Member, IEEE,* Jiang Xiao, *Member, IEEE,*

***Abstract*—A software defined network (SDN) enabled mobile access network is a future network with great potential to support scalable and flexible network applications. To support various network applications, the SDN-enabled mobile access network usually uses forwarding rules in SDN devices. With a limited rule space in existing SDN devices, a rule caching mechanism is an efficient way to improve the network performance. In this paper, we propose SRCMN, which is a new caching structure to improve the network performance with a limited rule space in the SDN-enabled mobile access network. We design a two-layer rule space in each SDN device, which is managed by the SDN controller. We also design a cache prefetching mechanism with the consideration of user mobility. By conducting extensive simulations, we demonstrate that our proposed structure and mechanism significantly outperform original rule space management under various network settings.**

***Index Terms*—SDN, mobile access network, Rule cache.**

I. Introduction

As a mobile access network evolves from 3G to 4G to accommodate dramatically increased mobile traffic, mobile access network infrastructure becomes increasingly chaotic and dense. Researchers are developing various new technologies to efficiently manage mobile access network resources [1]. A software defined network (SDN) has been proposed to offer scalable and flexible management with a logical centralized control model [2].

Mobile access network topology changes due to user mobility impose big challenges on rule management [3] which plays an important role in the SDN. When mobile users move after associated to base stations, they communicate with the Internet along different paths, on which the corresponding forwarding rules should be deployed in the backhaul of access networks. Meanwhile, each SDN-enabled switch stores forwarding rules in its local Ternary Content Addressable Memory (TCAM) which can compare an incoming packet to patterns in all of rules at a line rate simultaneously [4]. However, TCAM is not a cost-effective way to provide high performance with its expensive cost and high energy consumption. A limited TCAM size is a major restriction for adopting policies to support flow-based control in large-scale networks [5].

Rule placement optimization is an existing method to improve the processing capacity of data center networks [6]. The mechanism of this method is that by getting information on the whole network, after some analysis on all existing flows, a proper placement strategy can be found to improve the flow processing capacity. However, these optimizing strategies are usually static with a limited number of flows. Furthermore, when the status of flows changes, such as flow destination movement, traffic variation, etc., updating the rules in all switches is unfordable.

Mianxiong Dong, He Li and Kaoru Ota are with Muroran Insitute of Technology, Japan.

Jiang Xiao is with Hong Kong University of Science and Technology, Hong Kong.

Unlike rule placement optimization which regards a rule space as a limited resource, rule caching strategies use the space efficiently to store rules and cache the most frequently-used rules in TCAM as caching [3]. Therefore, all rules can be handled in the network via replacement policies and the performance can be enhanced in terms of high hit ratio. Compared to the rule placement optimization methods, the rule caching is a better approach to enable the flow-based control to provide both high performance and scalability, especially in a large-scale data center network.

In the mobile access network, another problem is that the user mobility changes the forwarding path of flows, which invalidates the cache strategies designed for the stable network connection. Therefore, except for the prediction of the flow traffic, the user mobility can be another issue to consider in mobile access networks. Since it is hard to predict the user mobility and a cache mechanism is online processing rather than traditional static rule placement, a mobile optimized prefetching algorithm is more suitable to mobile access networks.

To address these problems on rule management in SDN-enabled mobile access networks, in this paper, we set the TCAM as the rule caching and design a two-layer rule space to support a rule caching mechanism. With the consideration of user mobility, we also model the rule optimization problem and design a caching algorithm of prefetching and replacement strategies. This algorithm achieves high hit ratio by replacing rules to the flow forwarding paths and replacing them integrally with prediction of user mobility.

The rest of this paper are organized as follows. Section II reviews the SDN-enabled mobile access network and rule caching. Design of SRCMN and caching algorithm is introduced in Section III. Section IV gives results of the performance evaluation. Finally, Section V concludes this paper.

II. SDN-enabled Mobile Access Network and Rule Caching

The concept of SDN-enabled mobile access networks stems from the research of SDN and network virtualization provides a uniform programmatic interface for mobile access network management[7][8][9]. For the mobile access networks, SDN has the potential to make the most out of licensed network resources, provide better scalability for handling billions of users, and foster the innovation inside the network. As shown in Fig. 1, as well as the SDN-enabled wired network, a typical SDN-enabled mobile access network consists of the control

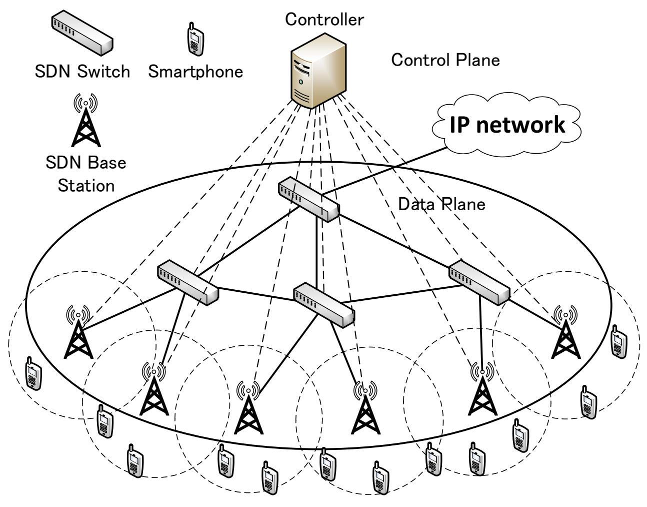


Fig. 1. Concept of the SDN-enabled mobile access network

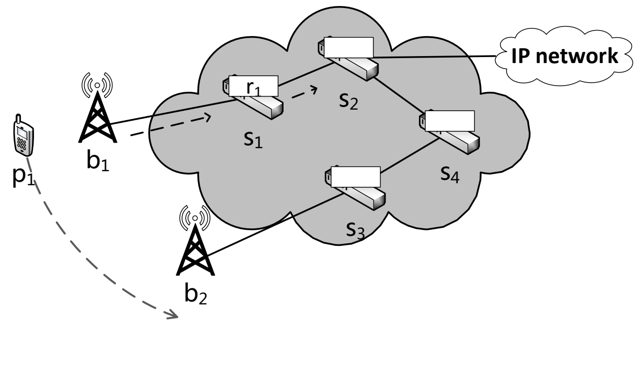
plane and the data plane. In the control plane, SDN devices, including base stations and switches, are connected in the network for forwarding network data between the IP network and mobile users. Each SDN device has a secure connection with the controller in the control plane.

In this network structure, the controller manages all network settings(e.g., frequency bands, forwarding strategies, etc.) of devices in the data plane. These settings abstract a set of rules and the controller installs these rules in corresponding devices. When a user smartphone sends a packet to the destination in the IP network, devices in the forwarding path will check their memory to find the matched rules for processing this packet. However, if the controller does not install the rules for this packet in the memory, the device will inform the controller for further processing.

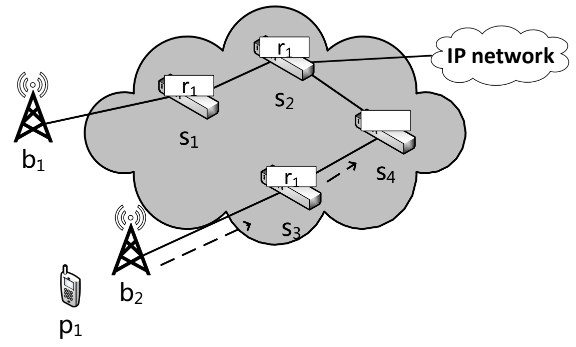
Since the connections between the control plane and devices are much slower than the connections in the data plane, the processing in the controller brings longer latency than in the devices. As a result, an ideal way to reduce this latency is installing all possible rules in the forwarding path and only processing packets in the controller when the network forwarding strategy is updated.

Unfortunately, it is hard to install too much rules in the SDN devices with the limitation of their memory. In most SDN devices, manufacturers use TCAM as the rule space to store rules with high matching speed. Since this type of memories brings very expansive cost and high energy consumption, rule space entries in most SDN devices are no more than 10 thousands. It is very strict to deploy comprehensive forwarding strategies in the mobile access networks with such few rules.

Rule caching is an efficient way to solve this issue that expands the rule space with normal memories instead of TCAM and uses TCAM as high-speed rule cache. Some previous work proposed practical switch structures [10][11][3][12][13] that uses a part of the memory in the switches as rule space and adds a caching algorithm to manage rule cache in the TCAM. However, since most design is based on a single device[14], existing work makes little consideration on the mobile access network in which device mobility brings more complex variation.



(a) User p_1 connects to base station b_1



(b) User p_1 moves to base station b_2

Fig. 2. Example of the SDN rule caching in the mobile access networks

III. SRCMN Design and Caching Algorithm

In this section, we first discuss the scenario of rule caching in SDN-enabled mobile access networks. Then, we describe the main structure and related modules in SRCMN structure. After that, we design a cache algorithm based on SRCMN structure to maximize the cache hit ratio.

A. Scenario analysis

For better understanding of the rule caching in SDN-enabled mobile access networks, we use an example to illustrate the scenario as shown in Fig. 2. There are two base stations b_1 and b_2 which connect to a mobile access network with four switches, s_1, s_2, s_3 and s_4. Mobile user p_1 moves from b_1 to b_2 and sends a network flow to the IP network. The small mobile access network uses rule r_1 to control this flow. Then, p_1 sends its flow from b_1 to switch s_1. Since this is the first time that the flow is transferred to s_1, the controller has to place rule r_1 to switch s_1. Usually, for the upcomming packets, this rule is placed in the cache of s_1 as shown in Fig. 2(a). When the flow is transferred to switch s_2, the controller needs to update rule r_1 to switch s_2 again and place r_1 to the cache, which means another cache miss.

In wired networks, after switch s_1 and s_2 cache rule r_1, there is no cache miss in transferring network flow of p_1.

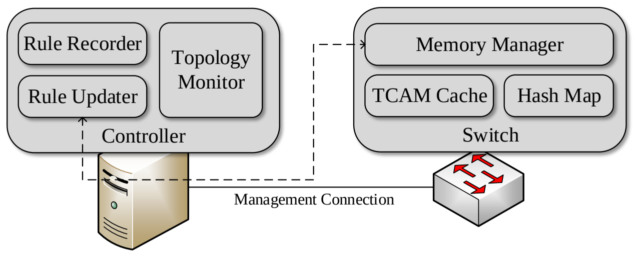


Fig. 3. Main modules in SRCMN include: Rule Recorder, Rule Updater and Topology Monitor in the controller; Memory Manager, TCAM Cache and Hash Map in all switches.

However, we focus on mobile access networks where users are mobile. As shown in Fig. 2(b), user p_1 moves to base station b_2 and the forwarding path is changed to s_3, s_4 and s_2. Therefore, when the flow comes to switch s_3 and s_4, the controller has to update rule r_1 and cache it to switches, which means two more cache misses.

As a result, during user p_1 moves from b_1 to b_2, four cache misses happen for only one rule r_1. In the rest of this paper, we will state this problem and propose a framework with a suitable cache algorithm. Meanwhile, the controller needs to update same rule r_1 four times, which is another problem in some previous work.

B. Rule Space Structure

For the problems of rule caching in mobile access networks, we design a two-layer rule space structure as shown in Fig. 3. In SRCMN, we design a memory manager as a module inserted in the existing programmable SDN devices and a cache manager as a network application deployed on the centralized controller. All rules are stored in SDN devices before the cache manager updates the rules. In the cache manager, we use three modules to support the cache mechanism: the rule recorder, the topology monitor, and the rule updater. The rule recorder stores all rules and the characters of their corresponding flows, while the rule updater sends updated information to the SDN-enabled switches. All topology information is stored in the topology monitor to support the decision of rule updating.

In the memory manager, we use a hash map to store the rules as the regular design in OpenvSwitch and a modified operating system in programmable SDN devices will access each rule entry missed in TCAM in this hash map as the second layer rule space. Meanwhile, the memory manager will update the cache miss information to the cache manager.

The rule updater receives the cache miss information and checks if this information causes cache updating. This updater checks the rule recorder and decides which rules need to be replaced by the rules for the new flows. How to make this decision is a problem dependent on the predicted flow traffic and the cache access history. With this updating decision, the rule updater predicts the next possible forwarding paths of the corresponding flows of this rules and prefetching them to these predict paths.

The rule recorder stores all rules both in existing network and network history. We also use a hash map to maintain these rules. We also maintain a table to store the characters of the corresponding flows of these rules. We choose the network predicted traffic and also the miss history as the main characters of each flow to support the decision of rule updating.

The topology monitor maintains information of each mobile users and the network topology. Meanwhile, this monitor also stores all forwarding paths for all base stations. For example, when a mobile user moves from one base station to another, the topology monitor can get the newest forwarding path as soon as possible.

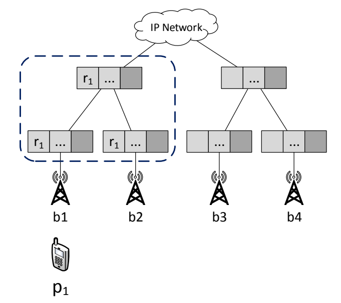
C. Cache Algorithm

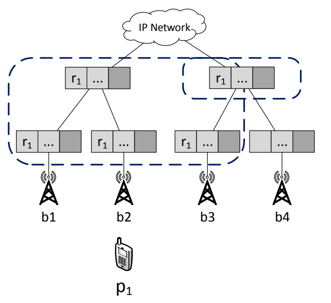
To maximize the cache hit ratio in SRCMN, we design a prefetching optimized *Least Recently Used* (LRU). The main improvement to the traditional LRU is adding a prefetching strategy with consideration of forwarding paths and user positions. We use an LRU queue in each switch to store the cached rules and the length of queue is the same with the TCAM size. The LRU queue is a basic data structure to maintain the least accessed rules. We add the prefetching strategy as an adjustment of LRU to suit the mobility and forwarding path, which is executed in the controller.

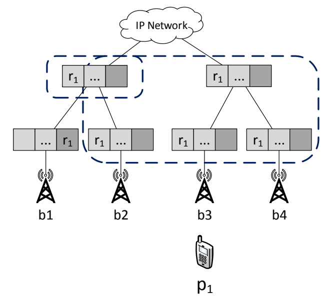
For the forwarding path, we first assume that the controller can place rules in all switches with similar time intervals. In SRCMN, we record all forwarding paths of each flow in the network. Therefore, we put rules in the corresponding switches in the whole forwarding path when the first switch in this path throws a rule miss event to the controller. Meanwhile, considering the neighboring base stations to usually share the same forwarding path, our prefetching strategy puts rules to a small number of switches.

In the prefetching strategy, considering the long access duration and the popular cellular access network [15], the next possible hops of mobility are predictable. As a result, since the neighbor base stations are limited with a prescribed network topology, we use an optimization that prefetches the rule to all neighbors. For example, in a regular cellular network, for each base station, the normal number of neighbors is six and we just need to prefetch rules in this base station and its six neighbors.

For better understanding of the caching algorithm, we use a simple example when a user moves from one base station to another base station. As shown in Fig. 4, we use a network topology with six switches and four base stations connected to this network. For each switch, we use some squares to illustrate LRU queues, in which the dark gray square means the tail of each queue. In this network, we put only one user p_1 who has flow f_1 to be forwarded to IP network. For forwarding this flow, we use r_1 as the forwarding rule. As shown in Fig. 4(a), user p_1 moves from base station b_1. Since there are two switches in the forwarding path from b_1 to IP network, the caching algorithm first puts r_1 in the head of the LRU queues of these two switches. Then, the algorithm prefetches this rule to the switches in the forwarding path of neighboring base


(a) User p_1 moves from base station b_1


(b) User p_1 moves to base station b_2


(c) User p_1 moves to base station b_3

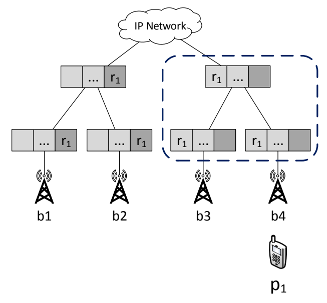

(d) User p_1 moves to base station b_4

Fig. 4. A simple example of the caching algorithm during user $p1$ moves from base station b_1 to b_4

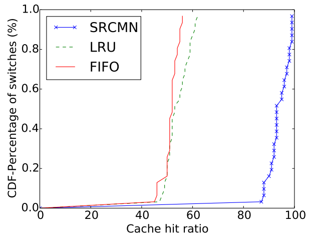
station b_2. Therefore, r_1 is put in the head of the LRU queues of three switches.

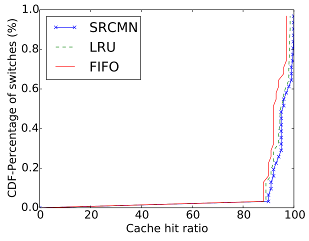
When the user moves to b_2 as shown in 4(b), since all the switches in the forwarding path of base station b_2 have cached rule r_1, there is no cache miss for this movement. For the prefetching, the algorithm puts r_1 in the head of the LRU queues of the switches in the forwarding path of neighboring base station b_3. After the user moves to b_3 as shown in 4(c), there is also no cache miss with the previous prefetching. Furthermore, since base station b_1 is not a neighbor of b_3, the algorithm moves r_1 to the tail of the LRU queues of the switch only in the forwarding path of b_1. When the user continues moving toward base station b_4, the algorithm puts r_1 to the tail of the LRU queues of the switches in the forwarding paths of base station b_2 as shown in Fig. 4(d).

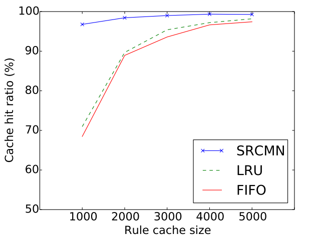
IV. Evaluation

In this section, we first evaluate the cache hit ratio by experiments on our prototype implementation. After that, we take a large scale simulation to evaluate the performance of SRCMN in a large mobile access network.

We have implemented the SRCMN framework as a network application on top of the popular OpenDaylight which is an open-source controller solution. We use mininet[16] as a main emulation tool and OpenvSwitch, an open source implementation of virtual OpenFlow switches. To reduce the overload of OpenDaylight, we implement a service outside the controller application to provide topology and placed rule information. The module in OpenDaylight is only used for the response to the caching missing event. For rapid development for the performance evaluation, we bypass the standard flow updating processing in openflow protocol and use a customized updating procedure to notify the switches to put rules in their cache.


(a) Small cache size


(b) Large cache size


(c) Slow movement environment

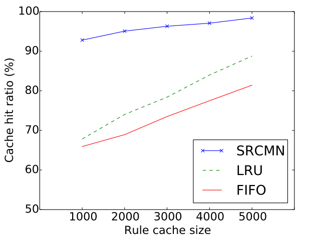

(d) Fast movement environment

Fig. 5. Evaluation results outperform the comparison algorithm in both micro benchmark and large scale simulations.

In each OpenvSwitch instance, for rule cache emulation, we have modified the original source code that changes the original rule space to limited flow entries and inserts a new structure to store infinite flow entries. Since we have no benchmark on the network performance, the additional rule space brings no latency than the original implementation. We use the original address mapping and change the address to the new flow entries when the number of flows exceeds the cache size.

We test one hour traffic and record the cache hit ratio of all switches in mininet. For comparison, we also use normal LRU and First In First Out (FIFO) with the same benchmark setting. We use two sizes of rule cache to find out the efficiency with small size cache and large size cache of SRCMN. Since our setting is a small scale network, the small size cache is set to 10 rule entries and the large cache size is set to 50 rule entries.

From the result with small size cache as shown in Fig. 5(a), the cache hit ratio of SRCMN is much better than other two cache algorithms with the small cache size. Comparing to the traditional LRU algorithm, even though the cache size with 10 rule entries is very small, SRCMN performs enough well with its prefetching strategy.

When the cache size becomes larger, the difference between these three cache algorithms becomes smaller as shown in Fig. 5(b) and SRCMN performs little better than other two algorithms. Considering there are only 30 flows in the network, the cache size with 50 rule entries is too large. As a result, SRCMN can improve the cache performance significantly with a limited cache size. Meanwhile, when the cache size is enough, all cache algorithms perform similar cache hit ratio even in mobility environment.

To evaluate the performance of the algorithm in SRCMN, we also use a simulation based benchmark to test the cache hit ratio with a large-scale network. We still use FIFO and LRU as comparison in this simulation. In the evaluation setting, we build a mobile access network with 1000 switches with a cache size from 1000 to 5000 rule entries. We also generate 10000 flows with mobility and set 500 base stations in the network which are distributed in the cellular structure. The hops from each base station to the IP network is still set to three and we still use the shortest paths as the forwarding paths. All flows have randomly distributed traffic from 1 bps to 10 Mbps. We test the cache hit ratio 20 times and get the average result.

For testing the different mobility of flows, we use two settings of movement speed: flows move from one base station to another in 20 minutes and 1 minute. We simulate 24 hours traffic and record the result of the cache hit ratio with two movement speed.

From the cache hit ratio result in the slow movement setting as shown in Fig. 5(c), SRCMN performs better than other two algorithms. Similar with the micro benchmark result, the small cache size influences the efficiency of FIFO and LRU seriously since these two algorithms have no optimization on the forwarding paths.

With faster movement speed, the cache hit ratio becomes worse as shown in Fig. 5(d), especially for FIFO and LRU. With the mobility optimized prefetching, SRCMN still performs efficiently with more than 90% cache hit ratio while other algorithms get near 80 % cache hit ratio even with the cache size of 5000 cache entries.

V. Conclusion

In this paper, we propose a rule caching model in the SDN-enabled mobile access network. With this model, we design a cache structure named SRCMN and apply the prefetching with the mobility and forwarding path of each flow to increase the cache hit ratio during users' migration in the network. We study a rule caching problem to maximize the cache hit ratio of the SDN-enabled mobile access network. Finally, micro benchmark and large scale simulations are conducted to show that the proposed caching algorithm can more significantly increase the hit ratio than traditional algorithms.

Acknowledgment

This work is partially supported by JSPS KAKENHI Grant Number 25880002, 26730056, JSPS A3 Foresight Program.

References

[1] M. Dong, T. Kimata, K. Sugiura, and K. Zettsu, "Quality-of-experience (QoE) in emerging mobile social networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 10, pp. 2606–2612, 2014.

[2] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. Reddy Sama, P. Seite, and S. Shanmugalingam, "An sdn-based network architecture for extremely dense wireless networks," in *Proceedings of IEEE SDN for Future Networks and Services (SDN4FNS 2013)*, Nov 2013, pp. 1–7.

[3] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the "one big switch" abstraction in software-defined networks," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*. New York, NY, USA: ACM, 2013, pp. 13–24.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[5] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 27–38.

[6] C. Meiners, A. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in tcams," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 488–500, April 2012.

[7] L. Li, Z. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Proceedings of European Workshop onSoftware Defined Networking (EWSDN 2012)*, Oct 2012, pp. 7–12.

[8] J. Kempf, B. Johansson, S. Pettersson, H. Luning, and T. Nilsson, "Moving the mobile evolved packet core to the cloud," in *Proceedings of IEEE 8th International Conference onWireless and Mobile Computing, Networking and Communications (WiMob 2012)*, Oct 2012, pp. 784–791.

[9] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 44–53, July 2013.

[10] Q. Dong, S. Banerjee, J. Wang, and D. Agrawal, "Wire speed packet classification without tcams: A few more registers (and a bit of logic) are enough," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '07)*. New York, NY, USA: ACM, 2007, pp. 253–264.

[11] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," in *Proceedings of the ACM SIGCOMM 2010 Conference (SIGCOMM '10)*. New York, NY, USA: ACM, 2010, pp. 351–362.

[12] G. Lu, R. Miao, Y. Xiong, and C. Guo, "Using cpu as a traffic co-processing unit in commodity switches," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*. New York, NY, USA: ACM, 2012, pp. 31–36.

[13] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Infinite cacheflow in software-defined networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN '14)*. New York, NY, USA: ACM, 2014, pp. 175–180.

[14] S. Luo, H. Yu, and L. M. Li, "Fast incremental flow table aggregation in sdn," in *Proceedings of the 23rd International Conference onComputer Communication and Networks (ICCCN 2014)*, Aug 2014, pp. 1–8.

[15] D. Astély, E. Dahlman, A. Furuskär, Y. Jading, M. Lindström, and S. Parkvall, "Lte: The evolution of mobile broadband," *IEEE Communicaiton Magazine*, vol. 47, no. 4, pp. 44–51, Apr. 2009.

[16] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. New York, NY, USA: ACM, 2010, pp. 19:1–19:6.