The Joint Optimization of Rules Allocation and Traffic Engineering in Software Defined Network

Huawei Huang*, Peng Li*, Song Guo*, and Baoliu Ye†
*School of Computer Science and Engineering, The University of Aizu, Japan
Email: {d8152101, pengli, sguo}@u-aizu.ac.jp
†Nanjing University, China
Email: yebl@nju.edu.cn

***Abstract*—Software-Defined Network (SDN) is a promising network paradigm that separates the control plane and data plane in the network. It has shown great advantages in simplifying network management such that new functions can be easily supported without physical access to the network switches. However, Ternary Content Addressable Memory (TCAM), as a critical hardware storing rules for high-speed packet processing in SDN-enabled devices, can be supplied to each device with very limited quantity because it is expensive and energy-consuming. To efficiently use TCAM resources, we propose a rule multiplexing scheme, in which the same set of rules deployed on each node apply to the whole flow of a session going through but towards different paths. Based on this scheme, we study the rule placement problem with the objective of minimizing rule space occupation for multiple unicast sessions under QoS constraints. We formulate the optimization problem jointly considering routing engineering and rule placement under both existing and our rule multiplexing schemes. Finally, extensive simulations are conducted to show that our proposals significantly outperform existing solutions.**

I. Introduction

Software-Defined Network (SDN) has been envisioned as the next generation network infrastructure, which promises to simplify network management by decoupling the control plane and data plane [1], [2]. In SDN, a centralized controller translates network management policies into packet forwarding rules, and deploys them to network devices, such as switches and routers. Each network device stores forwarding rules in its local Ternary Content Addressable Memory (TCAM) [3]–[5] that supports high-speed parallel lookup on wildcard patterns.

While TCAM excels in packet processing, it is an expensive hardware with high energy consumption. For example, it is reported that TCAMs are 400 times more expensive [5] and 100 times more power-consuming [6] per Mbit than RAM-based storage. As a result, each network device can be equipped with limited TCAM. However, the increasing demands would generate a large number of forwarding rules. The shortage of TCAM motivates us to investigate efficient rule placement in SDN such that as many traffic demands can be accommodated as possible.

In this paper, we consider a set of unicast sessions, each of which is associated with some endpoint policies between a source and a destination. These endpoint policies are translated into a set of forwarding rules that work as packet filters and should be applied to every packet from source to destination.

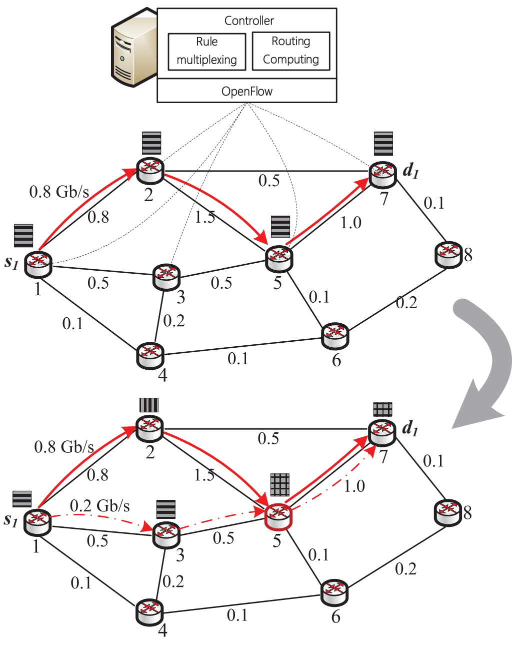


Fig. 1. Traffic engineering and rules multiplexing in SDN.

Each session specifies a throughput threshold to guarantee a certain level of Quality-of-Service (QoS).

Single-path routing has been widely used for unicast sessions because of its simplicity. However, it would be insufficient to satisfy the QoS requirement. For example, we consider a unicast session with 1Gb/s throughput requirement from source s_1 to destination d_1 in the network shown in Fig. 1, where the number on each link indicates its maximum transmission rate. Unfortunately, the best path $1 \to 2 \to 5 \to 7$ can achieve a throughput at most 0.8Gb/s. To achieve the required throughput, multiple paths can be employed for packet delivery. In Fig. 1, employing another path $1 \to 3 \to 5 \to 7$ with 0.2 Gb/s transmission rate simultaneously will achieve total throughput of 1 Gb/s. When multi-path routing is adopted in SDN, existing solutions [7]–[9] enforce endpoint policies by duplicating the same set of rules on each path of the session, leading to high TCAM consumption.

To deal with the TCAM-efficient rules placement in QoS-guaranteed multipath routing in SDN, we propose a rule multiplexing scheme. The basic idea is also explained by the motivation example in Fig. 1. Since node 5 is on the overlapped paths, a single copy of each rule that needs to be applied on this node is enough for the going-through traffic belonging to different paths. The above observation and further investigation lead to some new results that are summarized as follows.

- To extensively exploit the potential of our scheme, we study the rule placement problem with the objective of minimizing rule space occupation for multiple unicast sessions under QoS constraints.
- When a list of possible candidate paths for each session are given, we formulate the optimization problems under both existing and our rule multiplexing schemes.
- Finally, extensive simulations are conducted to show that our proposed rule multiplexing scheme can significantly reduce rule space occupation under QoS constraints.

The remainder of this paper is structured as follows. Section II reviews some related work in resource scheduling in SDN. Section III introduces the system model. Section IV proposes joint optimization to solve the rule placement with traffic engineering. The corresponding joint optimization problems with traffic engineering are investigated in Section V demonstrates the performance evaluation results by both case study and simulations. Finally, section VI concludes this paper.

II. Related Work

As the first attempt of building a network operating system at a large scale, NOX [10] achieves a simple programming model with control function based on OpenFlow. Recently, SDN-enabled switches and routers have been deployed in real large-sacle networks, such as Goolgle's G-scale network [11]. Ethane [12] has been proposed as a new network architecture for enterprises, which allows network managers to define a single network-wide fine-grained policy and then enforces it directly. Many existing work about SDN focuses on rule-space compression, rule split and distribution. DIFANE [7] and vCRIB [13] have been proposed to leverage all switches in a network to realize endpoint policies. Specifically, DIFANE uses a "rule split and caching" approach to increase the path length for the first packet of a flow. Later, Palette et al. [8] have proposed a framework for decomposing large SDN tables into small ones and then distributing them across the network, while preserving overall SDN policy semantics. Different from this work, we study joint routing and rule placement problems in this paper, which have never been investigated before. Kang et al. [9] have proposed a heuristic rule placement algorithms that distribute forwarding policies across general SDN networks while managing rule space constraints. Their solutions are obtained based on given routing scheme, while its effect on rule placement is ignored.

On the other hand, the multi-session multi-path QoS routing problem can be formulated as a Multi-Commodity Flow (M-CF) problem [14], [15]. Because a traffic flow can be split into sub-flows over multiple paths, each source-destination pair shall be expropriated to increase the packet forwarding

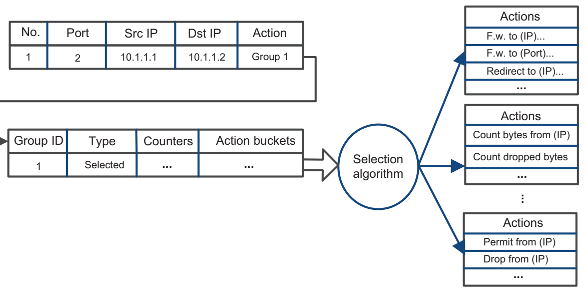


Fig. 2. Multi-path routing implementation in OpenFlow.

capability or to avoid congestion. In [16], a fundamental traffic engineering problem is studied to find minimum number of paths to achieve the maximum throughput. The effects of data center traffic characteristics on data center traffic engineering have been investigated in [17], where a system called MicroTE is developed to leverage the short term and partial predictability of the traffic matrix to adapt to traffic variations.

III. Network model

We model the SDN as a graph G=(N, E), where node set N consists of SDN-enabled network devices, and edge set E represents the communication links among these devices. Each device $u \in N$ maintains a TCAM-based flow table that can accommodate at most $\mathcal{C}_u$ rules. The bandwidth of each link $(u, v) \in E$ is denoted by $\mathcal{B}_{(u,v)}$.

We consider a set of K unicast sessions, and each session $k \in K$ imposes a QoS requirement with throughput $\mathcal{D}_k$ from a source s_k to destination d_k. A set L_k of candidate paths is given for each session $k \in K$. This scenario is practical in reality. For example, these candidate paths are pre-selected according to delay requirements. Furthermore, each session k is associated with a collection of rules (e.g., for access control, or network measurements). Usually, these rules cannot be accommodated by a single node due to limited TCAM capacity. To deploy these rules across the network, we use the algorithm proposed in [8] to decompose them into multiple subsets, which are maintained in I_k. Each subset $i \in I_k$ is an atomic unit with a number of c_i well-ordered rules that cannot be scattered over multiple nodes for the sake of semantic integrity. As a result, these rule subsets can be placed along routing paths in an arbitrary order.

In OpenFlow protocol, multi-path routing is implemented using group tables as shown in Fig. 2. When a packet is matched with a flow entry, we specify "group" action and process it with a group table. Typically, a group table consists of multiple group entries, each of which includes four fields as shown in Fig. 2. The group ID uniquely identifies a group, and counter is updated when packets are processed by a group. For multi-path routing, the field of type is set to "select", which means each packet will be processed by one of the action buckets stored in the last field. A user-defined selection algorithm determines which bucket will be applied for the

TABLE I
NOTATIONS

Notations	Description
N	a set of network devices
E	a set of links among devices
$\mathcal{C}_u$	TCAM capacity of node u
$\mathcal{B}_{(u,v)}$	bandwidth of link (u, v)
K	a set of unicast sessions
$\mathcal{D}_k$	throughput required by session k
I_k	a set of atomic rule subsets for session k
c_i	the number of rule in subset i, $\forall i \in I_k$
L_k	a set of paths for session k
x_u^i	a binary variable indicating whether rule set i is placed on node u
x_u^{il}	a binary variable indicating whether rule set i is on node u located in path l
y^l	a binary variable indicating whether path l is selected
r^l	a real variable representing the transmission rate on path l
$r_{(uv)}^l$	a real variable representing the transmission rate on link (u, v) along path l

packet.

In the traditional implementation, duplicated rules will be placed onto multiple buckets belonging to different paths such that the same set of endpoint policies will be executed along any path in the multi-path routing. This motivates us to reduce the rule space occupation by combining common rules among multiple buckets on each node. Finally, all symbols and variables used in the remaining of the paper are summarized in Table I.

All network and traffic demand information is maintained at the centralized controller that has a global view of the SDN. The objective of our rule placement problem is to minimize rule space occupation for a set of K unicast sessions under their QoS constraints.

IV. OPTIMIZATION WITH AVAILABLE PATHS

In this section, we consider to optimize rule space usage under a set of candidate paths. To solve this problem, we define a binary variable x_u^i for rule placement as follows:

$$x_u^i = \begin{cases} 1, & \text{if rule set } i \text{ is placed on node } u, \\ 0, & \text{otherwise.} \end{cases}$$

In addition, we define a binary variable x_u^{il} to describe the rule placement for each path:

$$x_u^{il} = \begin{cases} 1, & \text{if rule set } i \text{ is placed on node } u \text{ along path } l \\ 0, & \text{otherwise.} \end{cases}$$

Due to the rule multiplexing, each rule set placed at node u can be used by all paths going through it, leading to:

$$x_u^i = \max_{l \in L_k}\{x_u^{il}\}, \forall i \in I_k, \forall k \in K, \forall u \in N. \tag{1}$$

Note that only the rule sets belonging to the same session k can be multiplexed among paths in L_k. Since not all candidate paths need to be used for packet delivery, we define a binary variable y^l for path selection as follows:

$$y^l = \begin{cases} 1, & \text{if path } l \text{ is selected for packet delivery,} \\ 0, & \text{otherwise.} \end{cases}$$

By defining r^l and $r_{(u,v)}^l$ as the transmission rate on path l and link (u, v) on this path, respectively, the rule placement problem can be formulated as:

$$\min \sum_{k \in K} \sum_{i \in I_k} \sum_{u \in N} x_u^i,$$

$$\sum_{u \in l} x_u^{il} \geq y^l, \forall i \in I_k, \forall l \in L_k, \forall k \in K; \tag{2}$$

$$x_u^i \geq x_u^{il}, \forall l \in L_k, \forall i \in I_k, \forall k \in K; \tag{3}$$

$$\sum_{k \in K} \sum_{i \in I_k} x_u^i c_i \leq \mathcal{C}_u, \forall u \in N; \tag{4}$$

$$\sum_{l \in L_k} r^l \geq \mathcal{D}_k, \forall k \in K; \tag{5}$$

$$0 \leq r^l \leq r_{(u,v)}^l, \forall (u,v) \in l, \forall l \in L_k, \forall k \in K; \tag{6}$$

$$\sum_{k \in K} \sum_{l \in L_k} r_{(u,v)}^l \leq \mathcal{B}_{(u,v)}, \forall (u,v) \in E; \tag{7}$$

$$0 \leq r_{(u,v)}^l \leq y^l \mathcal{B}_{(u,v)}, \forall (u,v) \in E, \forall l \in L_k, \forall k \in K; \tag{8}$$

$$x_u^i, x_u^{il}, y^l \in \{1, 0\}.$$

As indicated in constraint (2), if a path $l \in L_k$ is selected, i.e., $y^l = 1$, each rule set $i \in I_k$ should be deployed on at least one node along this path, i.e., $\sum_{u \in l} x_u^{il} \geq 1$. Otherwise, i.e., $y^l = 0$, we do not constrain rule placement on this path, i.e., $\sum_{u \in l} x_u^{il} \geq 0$ that is always satisfied. The $\max$ operation in (1) can be replaced by (3). The number of rules placed at node $u \in N$ cannot exceed its rule space capacity as represented by (4). Constraint (5) guarantees the QoS of each session $k \in K$ by letting the total transmission rate of all selected paths be greater than $\mathcal{D}_k$. The transmission rate of a path is determined by the link with the minimum rate, which is represented by constraint (6). Constraint (7) represents that multiple paths associated with a common link should share the bandwidth of this link.

Although the above formulation is a mixed integer linear programming (MILP), there exists highly efficient algorithms, e.g., branch-and-bound, and fast off-shelf solvers, e.g., C-PLEX. Since our focus is to develop new schemes for rule placement and the corresponding optimization problems, we omit the details of solving MILP in this paper .

To better understand the benefits of our proposed rule multiplexing scheme, the same optimization problem under the traditional rule placement scheme is also formulated as follows.

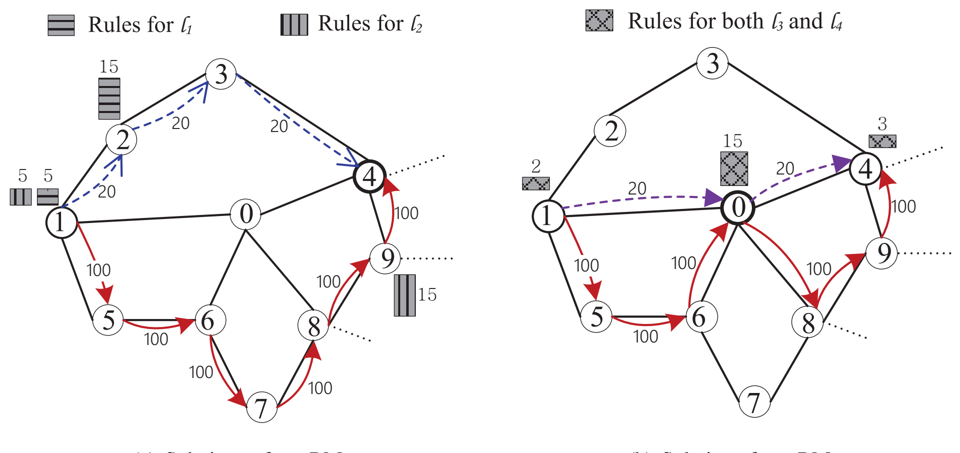
$$\min \sum_{k \in K} \sum_{i \in I_k} \sum_{l \in L_k} \sum_{u \in l} x_u^{il}$$

$$\sum_{k \in K} \sum_{i \in I_k} \sum_{l \in L_k} x_u^{il} c_i \leq \mathcal{C}_u, \forall u \in N; \tag{9}$$

$$(2), (5) - (8);$$

$$x_u^{il}, y^l \in \{0, 1\}.$$

Recall that the traditional scheme duplicates the same set of rules on each path of a session, resulting in that TCAM capacity constraint (4) is replaced by (9). Accordingly, its associated constraint (3) is also eliminated in above formulation.



(a) Solutions of nonRM (b) Solutions from RM

Fig. 3. Case study of two schemes.

V. Performance Evaluation

In this section, we first give a case study to demonstrate the operation law and advantages of our proposed mechanism, and then conduct extensive simulations to evaluate the performance of the proposed formulations on the ITALYNET network, which is widely used in literatures [18] and [19], where 20 datacenter nodes and 54 links are deployed. All mathematical programming formulations are solved using commercial solver Gurobi optimizer [20], and all simulation results are averaged over 100 instances. In the following, we use RM and nonRM to denote the schemes where rule multiplexing is applied or not, respectively.

A. *Case Study*

In the case study, we set the ITALYNET network (only partial) shown in Fig. 3 by fixing the capacity of link, the capacity of switch, the size of endpoint policy as 100, 15, 20, respectively. We consider a unicast session from node 1 to node 4 with a throughput requirement of 120 Mbit/s. There are four candidate paths, i.e., $l_1 = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$, $l_2 = \{1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 4\}$, $l_3 = \{1 \rightarrow 0 \rightarrow 4\}$ and $l_4 = \{1 \rightarrow 5 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 9 \rightarrow 4\}$.

Without rule multiplexing, paths l_1 and l_2 are selected for packet delivery as shown in Fig. 3(a). Since the rule space at source and destination is not enough to accommodate all rules, they are distributed on multiple nodes along the paths. For example, each path puts 5 rules at the source node, and the rest are placed at node 2 and node 9 on different paths, respectively. In that case, the total occupied rule space is 40.

When rule multiplexing is enabled, as shown in Fig. 3(b), we choose the paths l_3 and l_4 that share the rules placed at source, destination and the common node 0. As a result, the total rule space occupation is only 20.

B. *Simulation results of general cases*

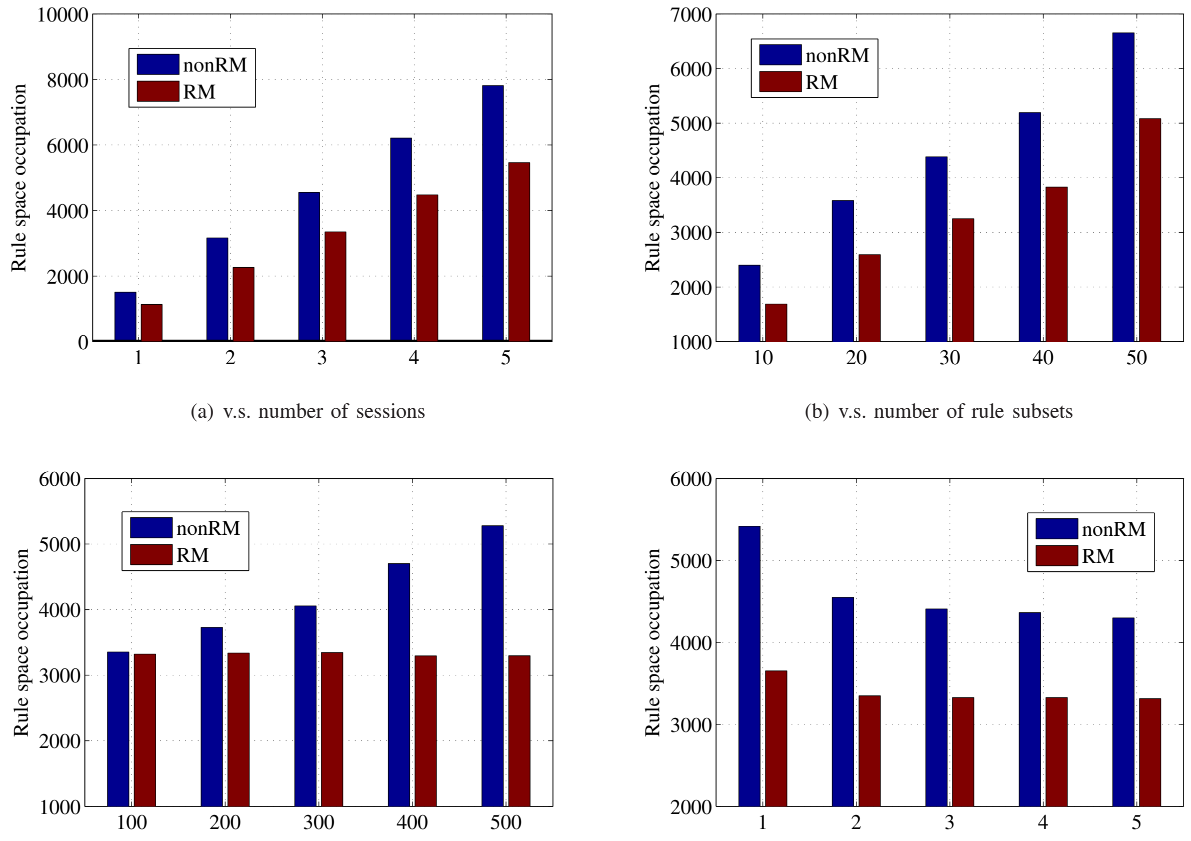
We then conduct extensive simulations on the complete ITALYNET topology to evaluate the performance of our proposed scheme. In a default network setting, the system parameters $B_{u,v}$, C_u, c_i and D_k are randomly generated according to uniform distributions within the ranges [150, 200], [1500, 2000], [10, 100] and [100, 200], respectively. The value of K, and the size of I_k are set to 3 and 20, respectively.

1) Rule space occupation: We first investigate the rule space occupation under different number of sessions. As shown in Fig. 4(a), rule space occupation increases linearly as the number of sessions grows under both schemes. That is because more paths are employed to achieve the QoS requirement, leading to more rule space occupation. Moreover, the performance gap between RM and nonRM becomes larger as the number of sessions increases. For example, when K=5, the rule space occupied by RM is less than 35% of nonRM.

We then study the effect of the size of I_k for by changing its upper bound from 10 to 50. The lower bound is fixed to 10. As shown in Fig. 4(b), although rule space occupation of both schemes show as increasing functions of the upper bound, RM always outperforms nonRM by saving much more rule space.

The rule space occupation under different QoS requirements is shown in Fig. 4(c). We change the upper bound of D_k from 100 to 500, and fix the upper bound of $B_{u,v}$ to 500. As illustrated in the figure, rule space occupation of nonRM grows as the increasing of required throughput, while the rules cost of RM is always around 3300. That is because rules can be multiplexed at the common nodes shared by multiple path under the RM scheme. It is worth noting that the performance of RM is only guaranteed within the range of bandwidth resource capacity. Once the QoS requirement exceeds the available bandwidth resources, the QoS can not be satisfied.

Finally, we change the number of candidate paths of each session from 1 to 5, and show the results in Fig. 4(d). We observe that less rule space is occupied when more candidate paths are provided under both RM and nonRM. Furthermore, the number of candidate path has little influence to the performance when it is larger than 3. This is because although



(a) v.s. number of sessions

(b) v.s. number of rule subsets

(c) v.s. maximum traffic rate

(d) v.s. indicator of path-set scale

Fig. 4. Rule space occupation of nonRM and RM.

more paths are available, they are not selected any more when the throughput requirement is satisfied of each session.

2) QoS satisfaction: We also study the QoS satisfaction, which is defined as the portion of sessions whose QoS requirements are successfully satisfied, under almost the same default settings aforementioned except that the bandwidth capacity of links and number of candidate paths are set to 500 and 1, respectively. We first investigate the influences of number of sessions on the QoS satisfaction. As shown in Fig. 5(a), the performance of both algorithms degrades as the number of sessions increases because larger number of traffic demands would quickly exhaust the bandwidth resources.

Then, we show the QoS satisfaction under different number of rules required by each session in Fig. 5(b), where RM outperforms nonRM. It clearly shows that the QoS satisfaction is a decreasing function of the number of rules.

We study the the effect of throughput requirements on the QoS satisfaction in Fig. 5(c). By randomly generating throughput requirements within different ranges, QoS satisfaction shows as decreasing functions under both schemes. The reason can be attributed to the fact that it becomes harder to guarantee QoS requirements under limited network bandwidth resources.

Finally, we investigate the QoS satisfaction under different switch capacity in Fig. 5(d). The performance of both schemes shows as increasing functions when switch capacity changes from 10 to 60. Note that after converging at switch capacity of 50, their performance is not affected by larger switch capacity.

VI. Conclusion

In this paper, we propose a rule multiplexing scheme, based on which a rule placement problem with the objective of minimizing rule space occupation for multiple unicast sessions under QoS constraints is studied. Then, we formulate the optimization problem jointly considering routing engineering and rule placement under both existing and our proposed rule placement schemes. Finally, extensive simulations are conducted to show that our proposals save TCAM resources significantly and guarantee high QoS satisfaction.

References

[1] F. Dürr, "Towards cloud-assisted software-defined networking," Technical Report 2012/04, Institute of Parallel and Distributed Systems, Universität Stuttgart, Tech. Rep., 2012.

[2] M. Mendonca, B. N. Astuto, X. N. Nguyen, K. Obraczka, T. Turletti *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," 2013.

[3] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.

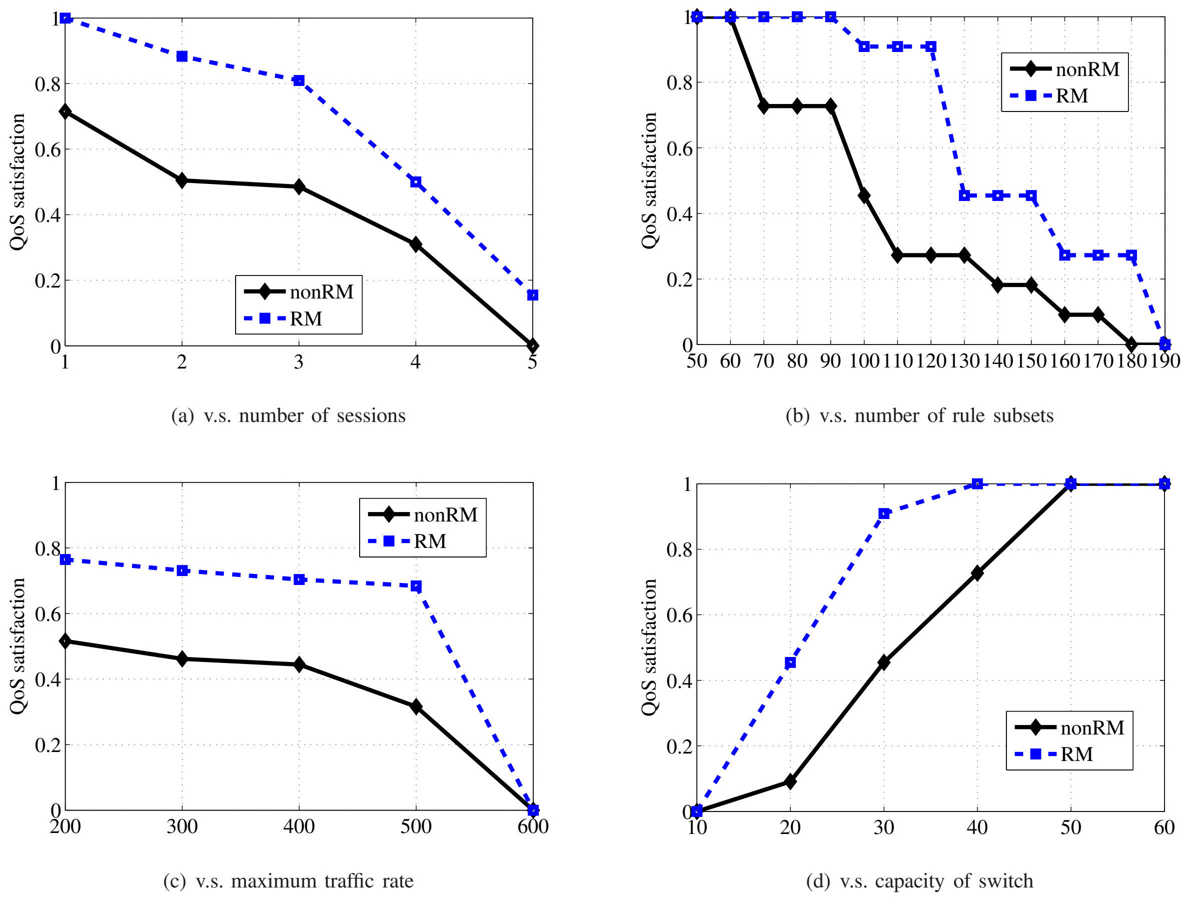


Fig. 5. QoS satisfaction of nonRM and RM.

[4] Y. Sun and M. S. Kim, "Tree-based minimization of tcam entries for packet classification," in *7th IEEE Consumer Communications and Networking Conference(CCNC),*. IEEE, 2010, pp. 1–5.

[5] N. Katta, J. Rexford, and D. Walker, "Infinite cacheflow in software-defined networks," Tech. Rep. TR-966-13, Department of Computer Science, Princeton University, Tech. Rep., 2013.

[6] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended tcams," in *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2003, pp. 120–131.

[7] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[8] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *IEEE INFOCOM*, 2013, pp. 545–549.

[9] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the one big switch abstraction in software-defined networks," *Proc. ACM CoNEXT*, 2013.

[10] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.

[11] "Google g-sacle network," https://www.eetimes.com/electronics-news/4371179/Google-describesits-OpenFlow-network.

[12] M. Casado, M. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, Aug 2009.

[13] M. Moshref, M. Yu, A. Sharma, and R. Govindan, "vcrib: virtualized rule management in the cloud," *HotCloud?2*, 2012.

[14] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM (JACM)*, vol. 37, no. 2, pp. 318–334, 1990.

[15] P.-O. Bauguion, W. Ben-Ameur, and E. Gourdin, "A new model for multicommodity flow problems, and a strongly polynomial algorithm for single-source maximum concurrent flow," *Electronic Notes in Discrete Mathematics*, vol. 41, pp. 311–318, 2013.

[16] H. Wang, J. Lou, Y. Chen, Y. Sun, and X. Shen, "Achieving maximum throughput with a minimum number of label switched paths in mpls networks," in *Proceedings of 14th International Conference onComputer Communications and Networks, (ICCCN)*. IEEE, 2005, pp. 187–192.

[17] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11, 2011, pp. 1–12.

[18] G. Sun, V. Anand, H.-F. Yu, D. Liao, and L. Li, "Optimal provisioning for elastic service oriented virtual network request in cloud computing," in *2012 IEEE International Conference on Global Communications Conference (GLOBECOM)*. IEEE, 2012, pp. 2517–2522.

[19] H. Huang, D. Zeng, S. Guo, and H. Yao, "Joint optimization of task mapping and routing for service provisioning in distributed datacenters," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1–6.

[20] G. Optimization, "Gurobi optimizer reference manual," 2013.