This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

Joint Optimization of Rule Placement and Traffic Engineering for QoS Provisioning in Software Defined Network

Huawei Huang, *Student Member, IEEE,* Song Guo, *Senior Member, IEEE,*
Peng Li, *Member, IEEE,* Baoliu Ye, *Member, IEEE,* and Ivan Stojmenovic, *Fellow, IEEE*

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. Via an extensive review of the state-of-the-art work, to the best of our knowledge, we are the first to study the non-routing-rule placement problem. Finally, extensive simulations are conducted to show that our proposals significantly outperform existing solutions.

Index Terms—Software-Defined Network, Ternary Content Addressable Memory, Rule Placement, Multipath Routing.

1 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, TCAM ternary match is 6 times expensive than Hash-based binary match in Static-RAM [6]. Further, it is reported that TCAMs are 400 times more expensive [5], [7] and 100 times more power-consuming [8] per Mbit than RAM-based storage. As a result, each network device can only be equipped with limited TCAM. Today's commodity switches typically support rule size from 2K to 20K only [9]–[11]. Additionally, the rule updating procedure in TCAM is quite slow, and about 40 to 50 rule updates per second [12], [13]. 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 traffic demands can be accommodated as many 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. 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 fail 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, in which even the best path $1 \rightarrow 2 \rightarrow 5 \rightarrow 7$ can achieve a throughput at most 0.8Gb/s. To achieve an imposed throughput, multiple paths can be employed for packet delivery. In the bottom case of Fig. 1, using another path $1 \rightarrow 3 \rightarrow 5 \rightarrow 7$ with 0.2 Gb/s transmission rate simultaneously will achieve total throughput of 1 Gb/s. However, when multipath routing is applied in SDN, existing solutions [10], [11], [14] 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 rule placement in QoS-guaranteed multipath routing, we propose a rule

- *H. Huang, S. Guo and P. Li are with the Performance Evaluation Lab, School of Computer Science and Engineering, The University of Aizu, Tsuruga, Ikki-machi, Aizu-Wakamastu City, Fukushima 965-8580, Japan. Email: {d8152101, sguo, pengli}@u-aizu.ac.jp.*
- *B. Ye is with the National Key Laboratory for Novel Software Technology, Nanjing University, Xianlin Campus, 163 Xianlin Avenue, Qixia District, Nanjing 210046, China. Email: yebl@nju.edu.cn.*
- *I. Stojmenovic is with the School of Information Technology and Engineering, University of Ottawa, Canada. E-mail: ivan@site.uottawa.ca*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

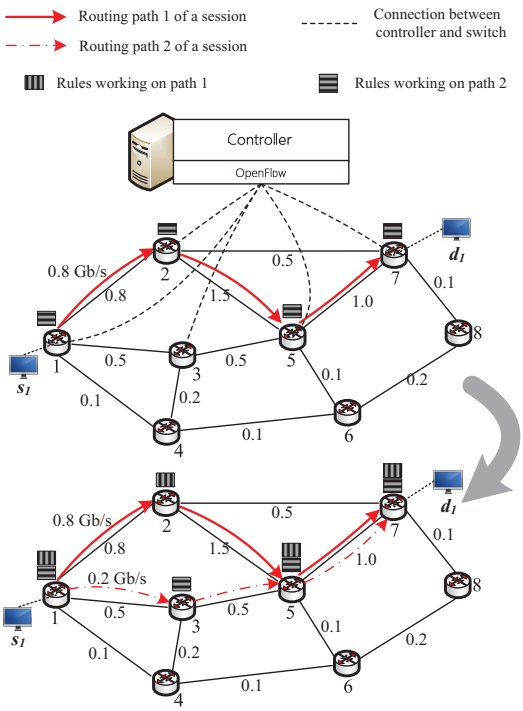


Fig. 1. The motivation case: traffic engineering and duplicated rules placement in traditional SDN enabled networks.

multiplexing scheme implemented in *Controllers*. As the example illustrated in Fig. 1, traditionally, the same set of rules should be deployed to both path 1 and path 2, denoted by the solid and dotted links from s_1 to d_1, respectively. In our proposed multiplexing scheme, only one copy of rules to be deployed onto the common nodes of multipath will be enough to manage entire going-through traffic belonging to different paths. For example, switches 1, 5, and 7, can jointly accommodate all rules in their TCAMs. We consider an example rule "`dl_src=`s_1, `dl_dst=`d_1 `actions=mod_vlan_vid`:0x0001", which modifies the VLAN IDs to 0x0001 for packets from source s_1 to d_1. Since packets along both paths share the same source and destination addresses, a single rule at switch 5 is enough to complete the VLAN ID modification.

In the rest of this paper, we let the abbreviation of `RM/nonRM` denote the scheme where rule multiplexing is applied or not, `CP/nonCP` indicate the scheme where candidate paths are provided or not. 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 non-routing-rules-oriented rules placement problem with the objective of minimizing rule space occupation for multiple unicast sessions under QoS constraints. We prove such optimization problem NP-hard.
- 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, i.e., the CP-based RM and nonRM optimizations.
- Then, we extend them to deal with more challenging scenarios that no candidate paths are provided by a joint optimizations of routing and rule placement, i.e., the nonCP-based RM and nonRM optimizations.
- Via an extensive review of the state-of-the-art work, to the best of our knowledge, we are the first to study the non-routing-rules-oriented RM-based rules placement problem.
- Finally, extensive simulations are conducted to show that our proposed rule multiplexing scheme can significantly reduce rule space occupation by approximately 10%∼35% under QoS constraints.

The remainder of this paper is structured as follows. Section 2 reviews some related work in resource scheduling in SDN. Section 3 discusses the system model and problem statement. Section 4 and Section 5 propose joint optimization to solve the rule placement and traffic engineering under the conditions with and without the given available paths set, respectively. Section 6 proposes a heuristic to solve the optimization of NP-hard problem. In performance evaluation, we firstly show a case study in Section 7. Then, Section 8 demonstrates the extensive simulation results of large-scale general topologies. Finally, Section 9 concludes this paper.

2 Related Work

2.1 SDN based networks

As the first attempt of building a network operating system at a large scale, NOX [15] achieves a simple programming model for control function based on OpenFlow. Later, Maestro [16] exploits parallelism with additional throughput optimization techniques while keeping the simple programming model for programmers. FlowVisor [17] is the first testbed for SDN, which slices the network hardware by placing a layer between control plane and the data plane. Its basic idea is that if unmodied hardware supports some basic primitives, then a worldwide testbed can ride on the coat-tails of deployments without extra expense. Recently, SDN-enabled switches and routers have been deployed in real large-sacle networks, such as Goolgle's G-scale network [18]. Ethane [19] has been proposed as a new network architecture for the enterprise, which allows managers to define a single network-wide fine-grained policy and then enforces it directly.

2.2 Rule space compression considering nonRM and RM

Many existing work about SDN focuses on rule-space compression, rule split and distribution. These work can be classified into two categories: 1) nonRM-based; 2) RM-based. A number of existing work [10], [11], [14], [20]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

belong to the first category. DIFANE [14] and vCRIB [20] 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. [11] have proposed a framework for decomposing large SDN tables into small ones and then distributing them across the network, while preserving the overall SDN policy semantics. Kang et al. [10] 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.

Different from references in the first category, we study the joint routing and rule multiplexing, i.e., the RM-based rules placement, problems in this paper, which have never been investigated before.

2.3 Multi-path routing considering nonCP and CP

The multi-session multi-path QoS routing problem can be also generally classified into two categories: nonCP based [21]–[25] and CP-based [26]–[29]. For example, Zhang et al. [21] have proposed routing optimization schemes to find a set of routes to minimize cost. In [22], a fundamental traffic engineering problem is studied to find minimum number of paths to achieve the maximum throughput. The effect of data center traffic characteristics on data center traffic engineering have been investigated in [23]. A system called MicroTE is developed to adapt to traffic variations by leveraging the short term and partial predictability of the traffic matrix. Nakibly et al. [24] have studied a problem of splitting traffic flow over multiple efficient paths to improve the network bandwidth utilization. However, using multiple paths for a traffic flow will increase the consumption of expensive forwarding resources, such as TCAM entries of switches and wavelengths of optical switches. They formulate and solve several problems of splitting a traffic flow over multiple paths while minimizing the overhead of forwarding resources. Agarwal et al. [25] have considered a scenario where SDN-enabled nodes are incrementally introduced into an existing network. They formulate an optimization problem with the objective of maximizing the network utilization. Furthermore, they propose fast algorithms to solve this problem with large-scale network instances.

The CP-based multi-path traffic engineering has been also extensively investigated. Wang et al. [26] have developed flow control algorithms for networks with multiple paths between each source-destination pair. Han et al. [27] have investigated the problem of congestion aware multi-path routing problem in the Internet by exploiting path diversity. Key et al. [28] have studied the benefits using multiple paths for a session with a joint consideration of rate control over paths and congestion control.

Different from the above work, our paper addresses the rule placement with both CP and non-CP cases.

3 Preliminary and Model

3.1 SDN rules

In OpenFlow specification [30], a flow table entry, i.e., SDN rule, consists of multiple matching and action fields. Once all conditions specified in the matching fields are satisfied, the corresponding actions specified in the action field will be executed by the host switch.

Some representative examples of matching fields are given as follows.

- `dl_src`: source data-link-layer (MAC) address of the packet
- `nw_dst`: destination network-layer address of the packet
- `dl_type`: protocol type of the packet
- `in_port`: incoming port number of the packet

In action field, the fundamental function is routing denoted by keyword `Output`. Other actions, e.g., `Set-queue`, `Drop`, `Push/Pop VLAN or MPLS Tag`, and `Set Field`, are more intensively applied to provide QoS support, secure access control, network management, and modification of packet header fields, respectively. These non-routing actions greatly improve the usefulness of OpenFlow implementations, e.g., network management, access control, and VLANs examples as reviewed in [31]. Since routing rules must be installed in each switch along the paths, we focus on the placement of non-routing rules in our study.

3.2 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 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 constrained 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. 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 [11] to decompose them into multiple subsets, which are maintained in I_k. Let $f(i)$ denote the session which a rule subset i belongs to. Each subset $i \subseteq I_{f(i)}$ is an atomic unit with a number of c_i well-ordered non-routing-oriented rules that cannot be scattered over multiple nodes for the sake of semantic integrity. As a result, these rule sets can be placed along the routing paths in an arbitrary order.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

TABLE 1
Notations

Notation	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)
$\bar{\mathcal{B}}$	$\max_{(u,v)\in E}\{\mathcal{B}_{(u,v)}\}$
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
$f(i)$	the mapping function of subset i to session k
c_i	the number of rules in subset i, $\forall i \subseteq I_k$, $k = f(i)$
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
$r^l_{(u,v)}$	a real variable representing the transmission rate on link (u, v) along path l
$\lambda^l_{(u,v)}$	a binary variable indicating whether link (u, v) is on path l

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.

3.3 Problem statement

All network and traffic demand information is maintained at the centralized controller that has a global view of the SDN.

With the given K unicast sessions and their traffic bandwidth requirements D_k, a set of candidate paths L_k can be selected for session k, a set of atomic rule subsets I_k for session k, we consider a rule placement problem with the objective to minimize the total rule space occupation for all sessions under their QoS constraints.

Theorem 1: Given a set of candidate paths, the rule placement problem mentioned above is NP-hard.

Proof: To prove an optimization problem to be NP-hard, we need to show the NP-completeness of its decision form, i.e., we attempt to find a rule placement such that the QoS of all sessions is satisfied, and total rule space occupation is no greater than X. It is easy to see that such a problem is in NP class as the objective associated with a given solution can be evaluated in a polynomial time.

The remaining proof is done by reducing the well-known 2-partition problem, i.e., given a set of numbers $A = \{a_1, a_2, ..., a_n\}$, we attempt to divide them into two sets such that $\sum_{j\in J_1} a_j = \sum_{j\in J_2} a_j = \mathcal{A}$, where J_1 and J_2 are index sets without overlapping. We now describe the reduction from the 2-partition problem to an instance

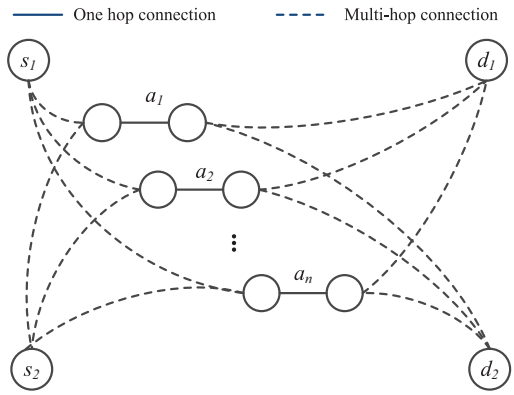


Fig. 2. Constructed instance of rule placement problem.

of our rule placement problem. We create two unicast sessions whose throughput should be no less than $\mathcal{A}$. The rule set of each session contains two rules. As shown in Fig. 2, for each number $a_j \in A$, we create two paths l and l' for both sessions, respectively, which share a bottleneck link of capacity a_j. Moreover, all nodes along these paths has no available rule space except the nodes associated with the bottleneck link, each of which can accommodate at most one rule. Finally, we let $X = 2n$;

In the following, we only need to show that the 2-partition problem has a solution if and only if the resulting instance of our rule placement problem has a solution that satisfies both QoS and rule space constraints. First, we suppose a solution of the 2-partition problem that the numbers can be divided into two sets with identical sum. The corresponding solution of our problem is to assign the paths of capacity $a_j, j \in J_1$ to one session, and the ones of capacity $a_j, j \in J_2$ to the other. It is easy to verify that the throughput of both session is $\mathcal{A}$, and the number of occupied rule space is X.

Then, we suppose that our rule placement problem has a solution with a total rule space X and throughput $\mathcal{A}$ for both sessions. Since only one rule can be accommodated at the nodes on the bottleneck link of each path, the two paths associated with a common bottleneck cannot be used by two sessions simultaneously. In order to achieve the throughput $\mathcal{A}$, the paths assigned to two sessions satisfy $\sum_{j\in J_1} a_j = \sum_{j\in J_2} a_j = \mathcal{A}$, which forms a solution of the 2-partition problem. □

4 Optimization with Candidate Paths

In this section, we consider to optimize the rule space usage when 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. To solve the rule placement problem, we define a binary variable x_u^i as follows:

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

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:

$$x_u^i = \max_{l \in L_k} \{x_u^{il}\}, \forall i \subseteq I_{k=f(i)}, \forall k \in K, \forall u \in N. \quad (1)$$

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:

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

If a path $l \in L_k$ is selected, i.e., $y^l = 1$, each rule set $i \subseteq I_{k=f(i)}$ should be deployed on at least one node along this path, i.e., $\sum_{u \in l} x_u^{il} \geq 1$. This constraint can be formulated as:

$$\sum_{u \in l} x_u^{il} \geq y^l, \forall i \subseteq I_{k=f(i)}, \forall l \in L_k, \forall k \in K. \quad (2)$$

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 the following equation:

$$x_u^i \geq x_u^{il}, \forall l \in L_k, \forall i \subseteq I_{k=f(i)}, \forall k \in K, \forall u \in N. \quad (3)$$

The number of rules placed at node $u \in N$ cannot exceed its rule space capacity as represented by:

$$\sum_{k \in K} \sum_{i \subseteq I_{f(i)}} x_u^i c_i \leq \mathcal{C}_u, \forall u \in N. \quad (4)$$

On the other hand, 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 QoS of each session $k \in K$ shall be guaranteed by letting the total transmission rate of all selected paths be greater than $\mathcal{D}_k$:

$$\sum_{l \in L_k} r^l \geq \mathcal{D}_k, \forall k \in K. \quad (5)$$

Furthermore, the transmission rate of a path is determined by the link with the minimum rate, which is represented by:

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

The characteristics of the association between routing paths and transmission rate should be specified. First, multiple paths associated with a common link should share the bandwidth of this link:

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

Then, the transmission rate on the link (u, v) in the selected path l shall between 0 and the maximum bandwidth of this link $\mathcal{B}_{(u,v)}$:

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

Finally, the multiplexing-considered rule placement problem with the objective minimizing the total allocated rule subsets under the candidate paths can be formulated as:

$$\min \sum_{k \in K} \sum_{i \in I_k} \sum_{u \in N} x_u^i c_i, \quad (9)$$
$$\text{s.t. : (2)} - \text{(8)};$$
$$x_u^i, x_u^{il}, y^l \in \{1, 0\}, r^l > 0, r_{(u,v)}^l > 0.$$

Although the above formulation (9) is a mixed integer linear programming (MILP), there exist highly efficient algorithms, e.g., branch-and-bound, and fast off-shelf solvers, e.g., CPLEX. 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.

$$\min \sum_{k \in K} \sum_{i \subseteq I_{k=f(i)}} \sum_{l \in L_k} \sum_{u \in l} x_u^{il} c_i \quad (10)$$

$$\sum_{k \in K} \sum_{i \subseteq I_{k=f(i)}} \sum_{l \in L_k} x_u^{il} c_i \leq \mathcal{C}_u, \forall u \in N; \quad (11)$$
$$\text{s.t. : (2), (5)} - \text{(8)};$$
$$x_u^{il}, y^l \in \{0, 1\}, r^l > 0, r_{(u,v)}^l > 0.$$

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 (11). Accordingly, its associated constraint (3) is also eliminated in above formulation (10).

5 Optimization without Candidate Paths

For many flow requests in practice, their candidate paths may not be specified by users, or constrained by any performance requirements (e.g., delay). When no candidate path is provided, the rule placement problem becomes more challenging but beneficial for TCAM-efficient QoS provisioning. To jointly consider traffic engineering and rule placement will raise the opportunities of both rule multiplexing and QoS guarantees. In this section, we investigate the rule placement problem without candidate path by developing a formulation that makes a good tradeoff between rule multiplexing and bandwidth utilization.

We define a binary variable $\lambda_{(u,v)}^l$ to indicate whether link (u, v) is selected by path l:

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

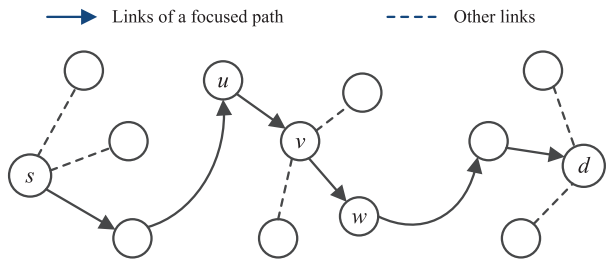


Fig. 3. An example for path searching.

The path searching process is represented by constraints (12) - (14).

$$\sum_{(s_k,v)\in E} \lambda^l_{(s_k,v)} - \sum_{(v,s_k\in E} \lambda^l_{(v,s_k} \leq 1, \quad \forall l \in L_k, \forall k \in K; \tag{12}$$

$$\sum_{(v,d_k\in E} \lambda^l_{(v,d_k} - \sum_{(d_k,v,)\in E} \lambda^l_{(d_k,v)} \leq 1, \quad \forall l \in L_k, \forall k \in K; \tag{13}$$

$$\sum_{(u,v)\in E} \lambda^l_{(u,v)} - \sum_{(v,w)\in E} \lambda^l_{(v,w)} = 0, \quad \forall v \in N \setminus \{s_k, d_k\}, \forall l \in L^k, \forall k \in K. \tag{14}$$

We use the example shown in Fig. 3 to explain these constraints, where solid arrows indicate a path from source s to destination d. At the source node, the number of outgoing links minus that of incoming links should be equal to 1 if this path is selected. Otherwise, their differences should be zero. A similar constraint (13) is imposed for the destination. At each intermediate node, for example, node v in Fig. 3, the number of incoming link should be equal to the number of outgoing link, which is represented by constraint (14).

In order to avoid cyclic paths, we particularly define an integer variable $z^l_{(u,v)}$ to denote the sequence number of the link along path l, i.e., (u,v) is the $z^l_{(u,v)}$-th link along the path $l \in L_k$ from the source s_k to the destination d_k. If link (u,v) is not on path l, i.e., $\lambda^l_{(u,v)} = 0$, its value of $z^l_{(u,v)}$ should be zero. Otherwise, the difference between the sequence numbers of two consecutive links on the path should be 1. Therefore, $z_{(u,v)}$ is between 0 and $|N|$-1 if link (u,v) is in path l:

$$0 \leq z^l_{(u,v)} \leq \lambda^l_{(u,v)}(|N|-1), \quad \forall (u,v) \in l, \forall l \in L_k, \forall k \in K. \tag{15}$$

$$\sum_{(v,w)\in E} z^l_{(v,w)} - \sum_{(u,v)\in E} z^l_{(u,v)} = \sum_{\forall (v,w)\in E} \lambda^l_{(v,w)}, \quad \forall v \in N - \{d_k\}, \forall l \in L_k, \forall k \in K. \tag{16}$$

With respect to rules placement, it is always shall be guaranteed that rules should be placed on and only on the nodes along the selected paths and shown as constraints (17) - (18):

$$x^{il}_u \leq \sum_{(u,v)} \lambda^l_{(u,v)} + \sum_{(v,d_k} \lambda^l_{(v,d_k}, \quad \forall i \subseteq I_{k=f(i)}, \forall l \in L_k, \forall k \in K, \forall u \in N; \tag{17}$$

$$\sum_{u\in N} x^{il}_u \geq \sum_{(s_k,v)} \lambda^l_{(s_k,v)}, \forall i \subseteq I_{k=f(i)}, \quad \forall l \in L_k, \forall k \in K. \tag{18}$$

The maximum transmission rate of each link (u,v) belonging to path l is constrained by $\mathcal{B}_{(u,v)}$ if this link is selected by l, i.e., $\lambda^l_{(u,v)} = 1$. Otherwise, $r^l_{(u,v)} = 0$. This can be described as:

$$0 \leq r^l_{(u,v)} \leq \lambda^l_{(u,v)}\mathcal{B}_{(u,v)}, \forall (u,v) \in E, \quad \forall l \in L_k, \forall k \in K. \tag{19}$$

Constraint (20) indicates that transmission rate of a path is determined by the bottleneck link. If a path (u,v) is on the path l, i.e., $\lambda^l_{(u,v)} = 1$, we get $0 \leq r^l \leq r^l_{(u,v)}$.

$$0 \leq r^l \leq r^l_{(u,v)} + (1 - \lambda^l_{(u,v)}\bar{\mathcal{B}}, \quad \forall (u,v) \in E, \forall l \in L_k, \forall k \in K. \tag{20}$$

Otherwise, constraint (20) becomes $0 \leq r^l \leq \bar{\mathcal{B}} = \max_{(u,v)\in E}\{\mathcal{B}_{(u,v)}\}$, which is always satisfied.

Finally, the relation between r^l and $\lambda^l_{(u,v)}$ can be specified as:

$$r^l \leq \sum_{(u,v)\in E} \lambda^l_{(u,v)}\bar{\mathcal{B}} \leq r^l \cdot \mathcal{M}, \quad \forall (u,v) \in E, \forall l \in L_k, \forall k \in K. \tag{21}$$

where $\mathcal{M}$ is an arbitrarily large number, such that all $\lambda^l_{(u,v)} = 0$, $\forall (u,v) \in l$, $\forall l \in L_k$ if $r^l = 0$.

Following the same definitions of x^i_u, x^{il}_u, r^l and $r^l_{(u,v)}$ in last section, the rule placement problem without candidate paths can be formulated as:

$$\min \sum_{k\in K} \sum_{i\subseteq I_{k=f(i)}} \sum_{u\in N} x^i_u c_i, \quad \text{s.t. : (3) – (5), and (7), (12) – (21);} \quad x^i_u, x^{il}_u, \lambda^l_{(u,v)} \in \{1,0\}, r^l > 0, r^l_{(u,v)} > 0. \tag{22}$$

The corresponding problem without rule multiplexing can be formulated as follows in a similar manner as given in last section.

$$\min \sum_{k\in K} \sum_{i\subseteq I_{k=f(i)}} \sum_{l\in L_k} \sum_{u\in l} x^{il}_u c_i \quad \text{s.t. : (5), (7), (11), (12) – (21);} \quad x^{il}_u, \lambda^l_{(u,v)} \in \{0,1\}, r^l > 0, r^l_{(u,v)} > 0. \tag{23}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

Algorithm 1 Fast Heuristic Algorithm

Input: Problem formulations with integer variables $x_u^i, x_u^{il}, \lambda_{(u,v)}^l \in \{0,1\}$

Output: Solutions $\widetilde{\lambda}_{(u,v)}^l, \widetilde{x}_u^{il}, \widetilde{x}_u^i$ of the original problem

1: obtain the solutions, i.e., $\widehat{x}_u^i, \widehat{x}_u^{il}, \widehat{\lambda}_{(u,v)}^l$, of optimization problems by relaxing all integer variables
2: **for all** $k \in K$ **do**
3: **for all** $l \in L_k$ **do**
4: $\widetilde{\lambda}_{(u,v)}^l \leftarrow$ PathSearch$(\widehat{\lambda}_{(u,v)}^l, k, l)$
5: $\widetilde{x}_u^{il} \leftarrow$ PathRulePlacement$(\widehat{x}_u^{il}, \widetilde{\lambda}_{(u,v)}^l, k, l)$
6: **end for**
7: $\widetilde{x}_u^i \leftarrow$ SessionRulePlacement$(\widehat{x}_u^i, \widetilde{x}_u^{il}, k)$
8: **end for**

Algorithm 2 PathSearch

Input: LP solution $\widehat{\lambda}_{(u,v)}^l$ $(\forall (u,v) \in E)$, session index k, path index l

Output: The rounded solutions $\widetilde{\lambda}_{(u,v)}^l$

1: $\widetilde{\lambda}_{(u,v)}^l \leftarrow 0, \forall (u,v) \in E$
2: Sort $Q = \{(u,v) | \widehat{\lambda}_{(u,v)}^l > 0\}$ as $\pi_1, \cdots, \pi_{|Q|}$ such that $\widehat{\lambda}_{\pi_1}^l \geq \cdots \geq \widehat{\lambda}_{\pi_{|Q|}}^l$
3: $P \leftarrow \emptyset$
4: **for** $j = 1; j \leq |Q|; j++$ **do**
5: $P \leftarrow P \cup \{\pi_j\}$
6: **if** $\lceil \widehat{\lambda}_{(u,v)}^l \rceil, \forall (u,v) \in P$ satisfy (12), (13) and (14) **then**
7: $\widetilde{\lambda}_{(u,v)}^l \leftarrow 1, \forall (u,v) \in P$
8: break
9: **end if**
10: **end for**

6 Heuristic Algorithms

Due to the NP-hardness of the rule placement problem, we propose a fast heuristic algorithm using relaxation and rounding techniques. As shown in Algorithm 1, we first solve the optimization problems by relaxing all integer variables, and then obtain feasible solutions by invoking PathSearch, PathRulePlacement, and SessionRulePlacement algorithms. Note that, with line 7, Algorithm 1 is `RM-nonCP` heuristic; otherwise, it becomes the `nonRM-nonCP` heuristic.

The pseudo codes of PathSearch algorithm is shown in Algorithm 2. All (u,v) tuples are sorted in a decreasing order according to values of $\widehat{\lambda}_{(u,v)}^l$ and are maintained in set Q. Then, we find feasible solutions satisfying constraints (12), (13) and (14) in the **for** loop from line 4 to 10.

Similarly, we find feasible solutions of x_u^{il} by first sorting them in a decreasing order in PathRulePlacement algorithm. All nodes belonging to the routes obtained from Algorithm 2 are maintained in set $\mathbb{V}$. Each element $\pi_j = (u', i')$ from Q is then checked sequentially, and is included in P if it satisfies the following conditions. 1) Node u' is in $\mathbb{V}$, 2) rule subset i' does not show in a (u,i)-tuple in P, and 3) the remaining space on node u' can accommodate rule subset i'.

Finally, the SessionRulePlacement algorithm is invoked to find feasible solutions of x_u^{il}. We first sort $\widehat{x}_u^i$

Algorithm 3 PathRulePlacement

Input: LP solution of $\widehat{x}_u^{il}$ $(\forall u \in N, \forall i, f(i) = k)$, $\widetilde{\lambda}_{(u,v)}^l$ $(\forall (u,v) \in E)$ from paths finding algorithm, k and l

Output: The rounded solutions $\widetilde{x}_u^{il}$

1: $\widetilde{x}_u^{il} \leftarrow 0$, $\forall u \in N, \forall i, f(i) = k$
2: Sort $Q = \{(u,i) | \widehat{x}_u^{il} > 0\}$ as $\pi_1, \cdots, \pi_{|Q|}$ in a decreasing order of $\widehat{x}_u^{il}$
3: $\mathbb{V} \leftarrow \{u, v | \widetilde{\lambda}_{(u,v)}^l = 1, \forall (u,v) \in E\}$
4: $P \leftarrow \emptyset$
5: **for** $j = 1; j \leq |Q|; j++$ **do**
6: $(u', i') \leftarrow \pi_j$
7: **if** $u' \in \mathbb{V}$ **and** $\forall (u,i) \in P, i' \neq i$ **and** $\sum_{\forall (u',i) \in P} c_i + c_{i'} \leq \mathcal{C}_{u'}$ **then**
8: $P \leftarrow P \cup \{\pi_j\}$
9: **if** $I_k \subseteq \bigcup_{\forall (u,i) \in P} \{i\}$ **then**
10: $\widetilde{x}_u^{il} \leftarrow 1$, $\forall (u,i) \in P$
11: break
12: **end if**
13: **end if**
14: **end for**

Algorithm 4 SessionRulePlacement

Input: LP solution of $\widehat{x}_u^i$ $(\forall u \in N, \forall i, f(i) = k)$, $\widetilde{x}_u^{il}$ $(\forall u \in N, \forall i, f(i) = k, \forall l, l \in L_k,)$ from Alg. 3, and k

Output: The rounded solutions $\widetilde{x}_u^i$

1: $\widetilde{x}_u^i \leftarrow 0$, $\forall u \in N, \forall i, f(i) = k$
2: Sort $Q = \{(u,i) | \widehat{x}_u^i > 0\}$ as $\pi_1, ..., \pi_{|Q|}$ in a decreasing order of $\widehat{x}_u^i$
3: $P \leftarrow \emptyset$
4: **for** $j = 1; j \leq |Q|; j++$ **do**
5: $(u', i') \leftarrow \{\pi_j\}$
6: **if** $(u', i') \in \{(u,i) | \widetilde{x}_u^{il} = 1\}$ **and** $\sum_{\forall (u',i) \in P} c_i + c_{i'} \leq \mathcal{C}_{u'}$ **then**
7: $P \leftarrow P \cup \{\pi_j\}$
8: **if** $I_k \subseteq \bigcup_{\forall (u,i) \in P} \{i\}$ **then**
9: $\widetilde{x}_u^i \leftarrow 1$, $\forall (u,i) \in P$
10: break
11: **end if**
12: **end if**
13: **end for**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

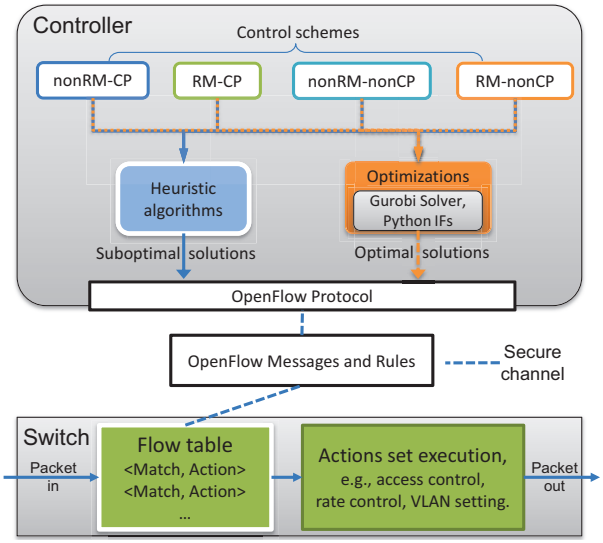


Fig. 4. The architecture of our simulation system.

in a decreasing order as shown in line 2, and then find feasible integer solutions in the following **for** loop.

Theorem 2: The time and space complexity of Algorithm 1 is $\mathcal{O}(|E| + \alpha\beta N + \alpha^2\beta N^2)$ and $\mathcal{O}(|N|\alpha(1+\beta) + |E|\beta)$, respectively, where $\alpha = \sum_{k=1}^{K} |I_k|$ and $\beta = \sum_{k=1}^{K} |L_k|$.

Proof: The worst computational complexity of Algorithms 2 and 3 is $\mathcal{O}(|\widehat{\lambda}^l_{(u,v)}|) = \mathcal{O}(|E|)$ and $\mathcal{O}(|\widehat{x}^{il}_u||\mathbb{V}|) = \mathcal{O}(N|I_k||L_k|)$, respectively. In addition, Algorithm 4 in line 7 has time complexity of $\mathcal{O}(\ |\widehat{x}^i_u||\widehat{x}^{il}_u|\ ) = \mathcal{O}(N^2 \cdot |I_k|^2 \cdot |L_k|)$. Therefore, the overall time complexity can be derived as follows.

$$
\begin{aligned}
&\mathcal{O}(Alg.p1)\\
&= \mathcal{O}(\sum_{k=1}^{K} |L_k| \times (\mathcal{O}(Alg.2) + \mathcal{O}(Alg.3))) + \sum_{k=1}^{K} \mathcal{O}(Alg.4)\\
&= \mathcal{O}(|E| + N \cdot \sum_{k=1}^{K} |I_k||L_k|) + \mathcal{O}(N^2(\sum_{k=1}^{K} |I_k|)^2 \sum_{k=1}^{K} |L_k|)\\
&= \mathcal{O}(|E| + \alpha\beta N + \alpha^2\beta N^2).
\end{aligned}
$$

The space complexity is determined by the number of variables x^u_i, x^{ul}_i, and $\lambda^l_{(u,v)}$, which can be calculated as $|N|\alpha$, $|N|\alpha\beta$, and $|E|\beta$, respectively. By summing up the number of all these variables and the corresponding rounded ones, the total space complexity is $\mathcal{O}(|N|\alpha(1+\beta) + |E|\beta)$. □

Note that, Algorithm 1 can be adopted when candidate paths are provided as a special case, in which variables $\lambda^l_{(u,v)}$ are fixed to one if $(u,v) \in l$ or to zero otherwise.

7 Case Study

7.1 Simulation settings

In this section, we demonstrate the rationale and advantages of the proposed mechanism by a case study on a partial ITALYNET topology [32], [33] as shown in Fig. 5, where link capacity is fixed to 100 and TCAM capacity of each switch is 15. Suppose we have a traffic demand from host h_1 (MAC address 00:00:00:00:00:01, ip address 10.0.0.1) to host h_2 (MAC address 00:00:00:00:00:02, ip address 10.0.0.2) with a bandwidth QoS request of 120. A set of provisioned non-routing-oriented rules, as illustrated in Table 2, need to be deployed on switches along each path. In our experimental results, we use the combinations of `RM/nonRM` and `CP/nonCP` to indicate the resulting schemes.

TABLE 2
20 SDN rules used in case study

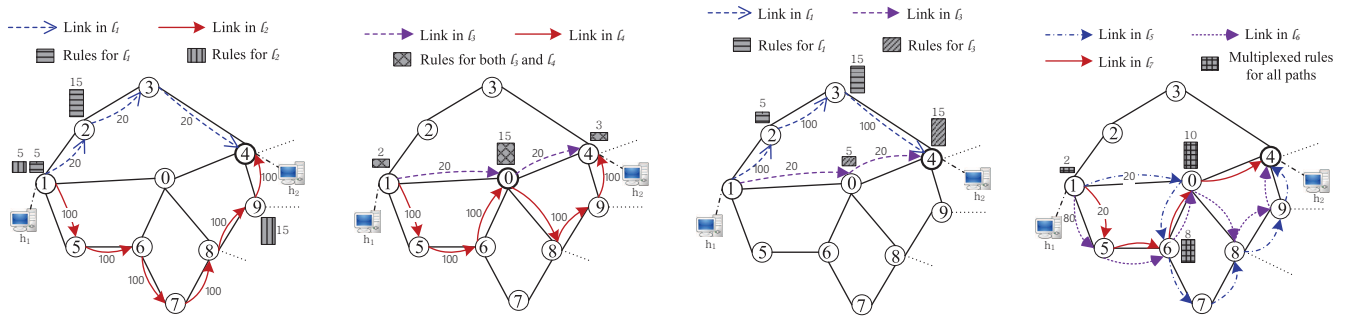
ID	Rules
1	add-flow sw_id dl_src=00:00:00:00:00:01, dl_dst=00:00:00:00:00:02,actions=mod_vlan_vid:0x0001
2	add-flow sw_id dl_src=00:00:00:00:00:01, dl_dst=00:00:00:00:00:02,actions=mod_nw_tos:0x10
3	add-flow sw_id nw_src=10.0.0.1,nw_dst=10.0.0.2, actions=drop
4	add-flow sw_id nw_src=100.0.0.3,actions=drop
5	add-flow sw_id dl_src=00:00:00:00:00:05,actions=drop
6	add-flow sw_id nw_ttl=53,actions=drop
7	add-flow sw_id dl_type=ipv6,actions=drop
8	add-flow sw_id dl_type=tcp6,actions=drop
9	add-flow sw_id dl_type=udp6,actions=drop
10	add-flow sw_id dl_type=icmp6,actions=drop
11	add-flow sw_id dl_type=ip,in_port=2,nw_src=10.0.0.4, actions=CONTROLLER:2024
12	add-flow sw_id dl_type=arp,in_port=3,arp_spa=10.0.0.4, actions=CONTROLLER:2025
13	add-flow sw_id dl_type=ip,in_port=1,nw_src=10.0.0.5, actions=CONTROLLER:2026
14	add-flow sw_id dl_type=ip,in_port=2,nw_src=10.0.0.6, actions=CONTROLLER:2027
15	add-flow sw_id dl_type=ip,actions=normal
16	add-flow sw_id dl_type=icmp,actions=normal
17	add-flow sw_id dl_type=tcp,actions=normal
18	add-flow sw_id dl_type=udp,actions=normal
19	add-flow sw_id dl_type=arp,actions=normal
20	add-flow sw_id dl_type=rarp,actions=normal

The internal architecture and relations between components of simulation are illustrated as Fig. 4. After solving optimization or heuristic algorithm, the obtained optimal or suboptimal solutions can be leveraged to place OpenFlow rules in data plane. We use Mininet as the emulator of data plane and Ryu as the OpenFlow controller. In rest of the paper, all mathematical programming formulations are solved using commercial solver Gurobi optimizer [34], which is embedded in the optimization program in terms of Python interfaces.

7.2 Solutions under given candidate paths

We first consider the problems when four available paths are provided, i.e., $l_1 = \{1 \to 2 \to 3 \to 4\}$, $l_2 = \{1 \to 5 \to 6 \to 7 \to 8 \to 9 \to 4\}$, $l_3 = \{1 \to 0 \to 4\}$ and $l_4 = \{1 \to 5 \to 6 \to 0 \to 8 \to 9 \to 4\}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers



(a) nonRM-CP (b) RM-CP (c) nonRM-nonCP (d) RM-nonCP

Fig. 5. Case study of four schemes with a 10-node scaled network, the rules are placed into the data plane according to the solutions obtained by solving four optimizations.

After solving the formulations of nonRM−CP and RM−CP, we display the solutions in Fig. 5(a) and 5(b). We observe that without rule multiplexing, paths l_1 and l_2 are selected for packet delivery as shown in Fig. 5(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 installs 5 rules at the source node, and the rest are placed at node 2 and node 9 on different paths, respectively. The total rules take 40 TCAM entries.

As shown in Fig. 5(b), when rule multiplexing is enabled, paths l_3 and l_4 are selected to share the rules placed at source, destination and the common node 0. As a result, the total rule space occupation is only 20.

7.3 Solutions without candidate paths

If the candidate paths are not provided, the optimal paths will be provided by solving the formulations of nonRM−nonCP and RM−nonCP. As a result, paths l_1 and l_3 are found in the solution of nonRM−nonCP. The resulting traffic distribution and rule placement along each path are illustrated in Fig. 5(c) with a total rule space occupation of 40.

Finally, Fig. 5(d) displays the solution from RM−nonCP. Three paths are calculated as $\{1 \rightarrow 5 \rightarrow 6 \rightarrow 0 \rightarrow 4\}$, $\{1 \rightarrow 0 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 4\}$ and $\{1 \rightarrow 5 \rightarrow 6 \rightarrow 0 \rightarrow 8 \rightarrow 9 \rightarrow 4\}$ with traffic rates of 20, 20 and 80, respectively. In particular, two rules (rule id: 12, 13) are placed on node 1, eight rules (rule id: 0, 3, 7, 10, 15, 16, 18, 19) on node 6, and ten rules (rule id: 1, 2, 4, 5, 6, 8, 11, 14, 17) on node 0. Such a placement guarantees that each path covers the whole rule sets.

8 Performance Evaluation

We have developed a simulation framework using C++ and realized the proposed heuristics using python. As shown in Fig. 4, the 4 rule placement schemes are embedded into our simulator, with which we conduct simulations to evaluate the performance of the proposed algorithms under various of network topologies. The demonstrated simulation result is averaged over 100 instances for each network setting.

8.1 Performance of the nonRM-CP and RM-CP

At the first, RM−CP and nonRM−CP schemes are evaluated with optimal solutions in a relative small scale of network, i.e., the ITALYNET network topology with 20 datacenter nodes, which has been widely used in literatures [32], [33]. The default settings of system parameters are as follows: $K = 3$, $|I_k| = 20$, $|L_k| = 2$, $\mathcal{B}_{(u,v)} \in [150, 200]$, $\mathcal{C}_u \in [1500, 2000]$, $c_i \in [10, 100]$, and $\mathcal{D}_k \in [100, 200]$, for $\forall k \in K, \forall (u, v) \in E, u \in N, i \in I_{f(i)}$.

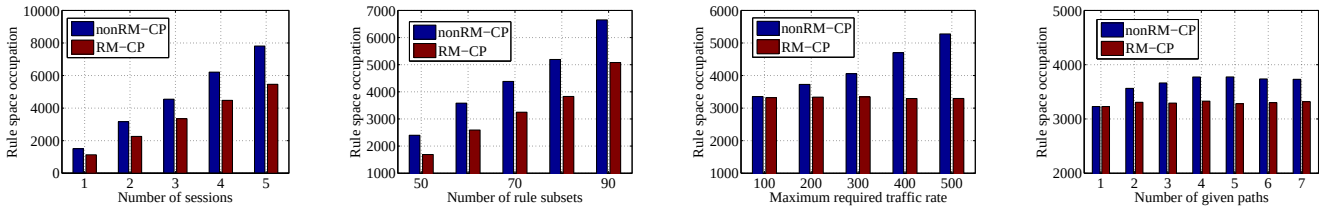
8.1.1 Rule space occupation

We first investigate the rule space occupation performance of our proposed rule placement schemes. By varying the number of sessions K from 1 to 5, Fig. 6(a) shows the occupied rule space increases linearly as the number of sessions grows because more paths are employed to achieve the throughput requirement, leading to more rule space occupation. The rule multiplexing scheme can save TCAM space significantly. For example, when K=5, the rule space needed by RM−CP is less than 35% of nonRM−CP.

Then, we study the effect of the size of I_k for each session by changing its upper bound value from 10 to 50 and fixing the lower bound as 10 when we generate its value randomly. As shown in Fig. 6(b), although the amounts of occupied rule space of both algorithms show as linear functions of the endpoint policy size, RM−CP outperforms nonRM−CP by saving about 30% of rule space utilization.

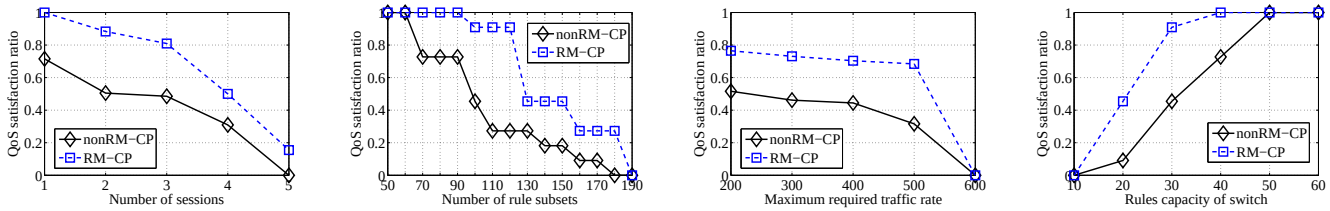
The performance under different maximum traffic rate requirements is shown in Fig. 6(c) by varying the upper bound of $\mathcal{D}_k$ from 100 to 500. Particularly, the upper range of $\mathcal{B}_{(u,v)}$ is reassigned to 500. As illustrated in this figure, the performance of nonRM−CP grows over the maximum traffic rate at a line rate, while the rules cost of RM−CP always holds around 3300. That is because rules can be always multiplexed at the nodes shared by multiple path with RM−CP scheme, no matter what the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers



(a) Rules cost v.s. number of sessions, i.e., K (b) Rules cost v.s. number of rule subsets, i.e., $|I_k|$ (c) Rules cost v.s. maximum traffic rate requirement, i.e., the upper bound of D_k (d) Rules cost v.s. number of given paths, i.e., $|L_k|$

Fig. 6. The optimal rule space occupation cost of nonRM−CP and RM−CP. This suite of simulations emphasize on comparing the performance of rule space occupation cost between nonRM and RM schemes, while providing the candidate paths.



(a) QoS satisfaction ratio v.s. number of sessions, i.e., K (b) QoS satisfaction ratio v.s. number of rule subsets, i.e., $|I_k|$ (c) QoS satisfaction ratio v.s. maximum traffic rate requirement D_k (d) QoS satisfaction ratio v.s. rules capacity of switch, $\mathcal{C}_u$

Fig. 7. QoS satisfaction ratio of nonRM−CP and RM−CP. This suite of simulations emphasize on comparing the performance of QoS satisfaction degree between nonRM and RM schemes, while providing the candidate paths.

traffic rate requirement is. However, it is worth noting that the performance of RM−CP is only guaranteed within the range of bandwidth resource capacity. Once traffic rate requirement exceeds the link bandwidth, the QoS can not be satisfied.

Later, we investigate how the size of available path set affects the performance. Note that, we use a modified weighted Dijkstra's algorithm to find available paths over the network graph for each session. In this algorithm, we iteratively update the available bandwidth of each link after finding the current shortest path until all required candidate paths are found. By setting the number of given paths for each session (i.e., $|L_k|, \forall k \in K$) from 1 to 7, and link bandwidth $\mathcal{B}_{(u,v)}$ in the range [150, 300], we show the rule space occupation cost of nonRM−CP scheme in Fig. 6(d). We notice that the two schemes performs the same when $|L_k|$=1, because single path cannot provide any opportunity for rule multiplexing. As $|L_k|$ grows, more rule space saving can be achieved by RM−CP over nonRM−CP, up to 30%. Such saving saturates soon when more candidate paths are given, e.g., $|L_k|$ >4. This can be attributed to the fact that RM−CP can always find some common nodes for rule multiplexing in a small number of candidate paths.

8.1.2 QoS satisfaction

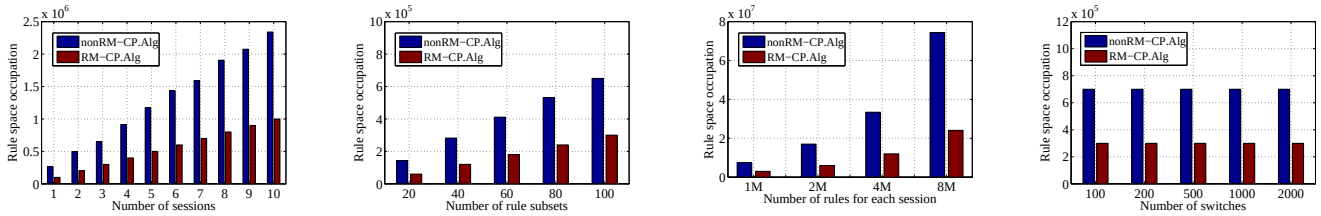
We also show the QoS satisfaction under link bandwidth of 500 and single-path routing. This metric is defined as the portion of simulation instances whose QoS requirements are satisfied. In each simulation case, since the traffic requirements from hosts are generated randomly, all the requirements possibly can not be guaranteed, i.e., it fails to find a feasible solution using the optimization of nonRM−CP or RM−CP with the given link bandwidth and routing path. Therefore, the simulation in section 8.1.2 is essentially the robustness comparison of nonRM-based and RM-based schemes under the given candidate paths.

We first investigate the influences of number of sessions on the QoS satisfaction. As shown in Fig. 7(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 scale of rule subsets of each session in Fig. 7(b), where RM−CP outperforms nonRM−CP. It clearly show the QoS satisfaction as a decreasing function over scale of rule subsets.

The effect of throughput requirements on QoS satisfaction is evaluated. As shown in Fig. 7(c), by randomly generating traffic rates within ranges [100,200], [200,300],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers



(a) Rule space cost v.s. number of sessions, i.e., K (b) Rule space cost v.s. number of rule subsets (c) Rule space cost v.s. number of rules for each session (d) Rule space cost v.s. number of switches, i.e., N

Fig. 8. Rule space occupation of fast heuristic algorithms under nonRM−CP and RM−CP schemes in randomly generated large-scale networks.

[300,400], [400,500], and [500,600], the QoS satisfaction shows a decreasing function of both schemes. The reason can be attributed to the fact that it becomes harder to guarantee all the requirements with the available resources under higher QoS requirement.

At last, we see the QoS satisfaction under different scale of switch capacity in Fig. 7(d) showing as increasing functions, when switch capacity varys within 10 and 60. After converging at switch capacity of 50, their performance is not affected by larger switch capacity.

Overall, from all the figures above we can always observe that the RM−CP significantly outperforms nonRM−CP of QoS satisfaction.

8.1.3 Performance in Large-Scale Networks

Then, we also evaluate the fast heuristics under nonRM−CP and RM−CP schemes in large-scale networks. The topology is randomly generated in each running case. The default settings of system parameters are as follows: $N = 500$, $K = 3$, $|I_k| = 100$, $|L_k| \in [2, 10]$, $\mathcal{B}_{(u,v)} = 100$, $\mathcal{C}_u = 800$K (K=10^3), $c_i = 1$K, and $\mathcal{D}_k \in [80, 300]$, for $\forall k \in K, \forall (u, v) \in E, u \in N, i \in I_{f(i)}$.

As shown in Fig. 8(a), we first evaluate the performance under various numbers of sessions by varying K from 1 to 10. The cost of rule space occupation linearly increases over K for both schemes. By extending $|I_k|$ from 20 to 100, Fig. 8(b) shows the rule space occupation cost is an increasing function of the number of rule subsets for each session. It can be also observed that RM−CP outperforms nonRM−CP by around 60% of the total rule space occupation. Then, we study the performance under various numbers of required rules for each session by varying $|I_k| \times c_i \in \{1\text{M}, 2\text{M}, 4\text{M}, 8\text{M}\}$ (M=10^6) and reassigning $\mathcal{C}_u = 4$M. Fig. 8(c) shows the rule space cost increases over the number of rules. Finally, we evaluate the performance under various scales of networks by varying the number of switches, i.e., $N \in \{100, 200, 500, 1000, 2000\}$. The rules occupation cost is shown in Fig. 8(d). It can be observed that the cost is little affected by the size of networks for both the nonRM−CP and RM−CP schemes. This is because although the provided candidate short paths are found a little different in various scales of networks, the total required number of paths is always the same when the required traffic rate of each session is fixed. As a result, the induced rule space occupation cost maintains similar in different running cases. Furthermore, we see that RM−CP shows much more efficient than nonRM−CP again.

8.2 Performance of the nonRM-nonCP and RM-nonCP

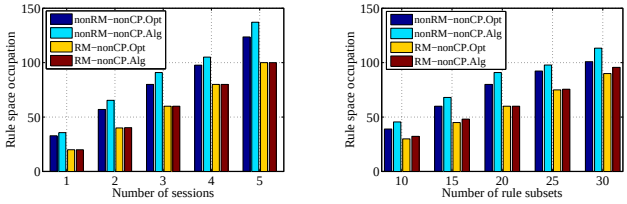
We firstly evaluate the performance of the proposed heuristic Alg. 1 under schems nonRM−nonCP and RM−nonCP with the corresponding optimal solutions in small-scale network, e.g., the same network topology adopted in Section 7, in which parameters are set as: $N = 10$, $|I_k| = 20$, $c_i = 1$, $\mathcal{B}_{(u,v)} \in [80, 200]$, $\mathcal{C}_u \in [15, 20]$ and $\mathcal{D}_k \in [100, 200]$.

By varying K from 1 to 5, Fig. 9(a) shows the comparison between the optimal rule occupation and the result obtained by applying Alg. 1. Their performance results as a function of $|I_k|$ in the range from 10 to 30 are also compared in Fig. 9(b). As we observe from both figures, the proposed Alg. 1 incurs only a little extra rule occupation, i.e., within 10% and 5% compared to the optimal solutions of nonRM−nonCP and RM−nonCP, respectively.

Then, extensive simulation experiments are conducted to show the performance of the heuristic algorithm under schems nonRM−nonCP and RM−nonCP in 30-node networks that are randomly generated by linking any two nodes with probability of 0.2. Since the corresponding optimal solutions can be obtained in a timely manner, we show the optimal solution of former two schemes, i.e., nonRM−CP and RM−CP, instead for the purpose of comparison. The default settings of simulation parameters are as follows: $|I_k| = 20$, $\mathcal{B}_{(u,v)} \in [100, 200]$, $\mathcal{C}_u \in [1500, 2000]$, $c_i \in [10, 100]$, and $\mathcal{D}_k \in [100, 200]$.

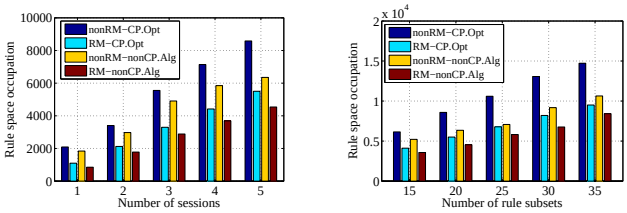
Fig. 10(a) shows the performance results of all four models under various number of sessions from 1 to 5. Both nonRM−nonCP and RM−nonCP models search their best routes for each session if available. For the models nonRM−CP and nonRM−nonCP that both apply the traditional rule placement mechanism, the latter can always obtain some improved performance. We

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers



(a) Rules cost v.s. K (b) Rules cost v.s. $|I_k|$

Fig. 9. Rule space occupation of `nonRM−nonCP` and `RM−nonCP` under a partial ITALYNET networks with 10 nodes. This suite of simulations emphasize on comparing the performance between Alg. 1 and optimal solutions.



(a) Rules cost v.s. K (b) Rules cost v.s. $|I_k|$

Fig. 10. Rule space occupation of `nonRM−nonCP` and `RM−nonCP` under randomly generated networks with 30 nodes. This suite of simulations emphasize on comparing the performance between nonRM and RM schemes, under the cases of CP and nonCP, respectively.

attribute it to the gain achieved by jointly optimizing multi-path routing and rule placement. When our proposed rule multiplexing scheme is applied, the corresponding models `RM−CP` and `RM−nonCP` achieve significant performance improvement over `nonRM−CP` and `nonRM−nonCP`, respectively. However, when comparing `RM−nonCP` to `RM−CP`, we notice only slight improvement achieved. This shows that the rule multiplexing scheme is sometimes more efficient to improve performance, especially when the given candidate paths are good enough already.

Finally, we show the experimental results in Fig. 10(b) when varying the number of rule subsets from 15 to 35 and fixing the number of unicast sessions to 5. The total rule space occupation of all models shows as an increasing function of rule subset sizes as explained in Section 8.1. Some other findings similar to the ones shown in Fig. 10(a) are also made. In summary, the advantage of our rule multiplexing mechanism can be always observed that the RM scheme achieves 30% less rules cost than nonRM scheme under CP case, while this quantitative improvements are approximately 10%∼20% under nonCP case.

9 Conclusion and Future Work

In this paper, we propose a rule multiplexing scheme for rule placement with the objective of minimizing rule space occupation for multiple unicast sessions under QoS constraints. We formulate an optimization problem by jointly considering routing engineering (i.e., with or without the given candidate paths) and rule placement under both the existing nonRM-based and our proposed RM-based rule placement schemes. Due to the NP-hardness, we propose heuristic algorithms for minimization problems of `RM−nonCP` and `nonRM−nonCP` using the relaxation and rounding techniques. Two phases are included in the major heuristic algorithm. In the first phase, we select multiple paths for each session, while rules placement solution is found at the selected routing paths in the second phase. The computational complexity is also analyzed. Finally, extensive simulations are conducted to show that our proposals and heuristic algorithms save TCAM resources significantly. Our future work includes the derivation of approximation ratio of our heuristic algorithm.

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.

[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] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 99–110.

[7] "Sdn system performance," http://pica8.org/blogs/?p=201, 2012.

[8] 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.

[9] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "Past: Scalable ethernet for data centers," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT)*. ACM, 2012, pp. 49–60.

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

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

[12] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 539–550.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

[13] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN)*. ACM, 2013, pp. 43–48.

[14] 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.

[15] 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.

[16] Z. Cai, A. L. Cox, and T. E. N. Maestro, "A system for scalable openflow control," Technical Report TR10-08, Rice University, Tech. Rep., 2010.

[17] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

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

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

[20] M. Moshref, M. Yu, A. Sharma, and R. Govindan, "Vcrib: Virtualized rule management in the cloud," in *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing, (HotCloud'12)*, 2012.

[21] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Towsley, "On optimal routing with multiple traffic matrices," in *Proceedings of IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM)*, vol. 1. IEEE, 2005, pp. 607–618.

[22] 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.

[23] 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.

[24] G. Nakibly, R. Cohen, and L. Katzir, "Optimizing data plane resources for multipath flows," *IEEE/ACM Transactions on Networking (TON)*, vol. PP, no. 99, 12 2013.

[25] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE Proceedings of IEEE 32th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, April 2013, pp. 2211–2219.

[26] W.-H. Wang, M. Palaniswami, and S. H. Low, "Optimal flow control and routing in multi-path networks," *Performance Evaluation*, vol. 52, no. 2, pp. 119–132, 2003.

[27] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 6, pp. 1260–1271, 2006.

[28] P. Key, L. Massoulié, and D. Towsley, "Path selection and multipath congestion control," in *26th IEEE International Conference on Computer Communications, (INFOCOM)*. IEEE, 2007, pp. 143–151.

[29] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: performance issues and a possible solution," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1651–1665, 2013.

[30] "Openflow specification," Open Networking Foundation, 2013.

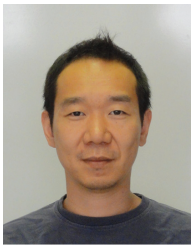
[31] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[32] 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.

[33] 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.

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

Huawei Huang received the Master degree in computer science from the China University of Geoscience (Wuhan) in 2013. He is currently a PhD candidate at School of Computer Science and Engineering, The University of Aizu, Japan. His research interests are mainly in the area of software defined networking and wireless networks.



Song Guo (M'02-SM'11) received the PhD degree in computer science from the University of Ottawa, Canada in 2005. He is currently a Full Professor at School of Computer Science and Engineering, the University of Aizu, Japan. His research interests are mainly in the areas of protocol design and performance analysis for wireless networks and distribted systems. He has published over 250 papers in refereed journals and conferences in these areas and received three IEEE/ACM best paper awards. Dr. Guo currently serves as Associate Editor of IEEE Transactions on Parallel and Distributed Systems, Associate Editor of IEEE Transactions on Emerging Topics in Computing with duties on emerging paradigms in computational communication systems, and on editorial boards of many others. He has also been in organizing and technical committees of numerous international conferences. Dr. Guo is a senior member of the IEEE and the ACM.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2401031, IEEE Transactions on Computers

Peng Li received his BS degree from Huazhong University of Science and Technology, China, in 2007, the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. He is currently an Associate Professor in the University of Aizu, Japan. His research interests include networking modeling, cross-layer optimization, network coding, cooperative communications, cloud computing, smart grid, performance evaluation of wireless and mobile networks for reliable, energy-efficient, and cost-effective communications. He is a member of IEEE.

Baoliu Ye received his PhD degree in computer science from Nanjing University, China in 2004. He is now an associate professor at the department of Computer Science and Technology, Nanjing University, China. He served as a visiting researcher of the University of Aizu, Japan from March 2005 to July 2006. His current research interests include Peer-to-Peer (P2P) computing, online/mobile social networking, and wireless network. He has published over 40 technical papers in the above areas. He served as the TPC co-chair of HotPOST'12, HotPOST'11, P2PNet'10. He is the regent of CCF and a member of IEEE, ACM.

Ivan Stojmenovic was editor-in-chief of IEEE Transactions on Parallel and Distributed Systems (2010-3), is Associate Editor-in-Chief of Tsinghua Journal of Science and Technology, and is founder of four journals (Journal of Multiple Valued Logic and Soft Computing, Ad Hoc & Sensor Wireless Networks, International Journal of Parallel, Emergent and Distributed Systems, Cyber Physical Systems). He is editor of IEEE Transactions on Computers, IEEE Network, IEEE Transactions on Cloud Computing etc. Stojmenovic has top h-index in Canada for mathematics and statistics, and has $>$16000 citations and h=63. He received five best paper awards. He is Fellow of IEEE, Canadian Academy of Engineering and Academia Europaea. He received Humboldt Research Award in Germany and Royal Society Research Merit Award in UK. He is Tsinghua 1000 Plan Distinguished Professor.