Load-Balancing Software-Defined Networking Through Hybrid Routing

Gongming Zhao, Liusheng Huang(✉), Ziqiang Li, and Hongli Xu

School of Computer Science and Technology,
University of Science and Technology of China, Hefei, Anhui 230027, China
{zgm1993,lzqrush}@mail.ustc.edu.cn, {lshuang,xuhongli}@ustc.edu.cn

Abstract. In recent years, software defined network (SDN) has become a promising technology to improve the network utilization. However, due to the limited flow table size and long deployment delay, it may result in low network performance in datacenter networks, which makes the user experience much worse. In this paper, we first propose a novel Tag-based Rule Placement Scheme (TRPS) for wildcard routing. Then, we study the Hybrid Routing problem by Joint Optimization of Per-Flow Routing and Tag-based Routing (HR-JPT). Besides, an approximation algorithm (RRJD) is proposed to solve the HR-JPT problem. The simulation results on Mininet platform show that our proposed algorithms are efficient for software-defined networking.

Keywords: SDN · Load balancing · Per-flow routing · Tag-based routing · Hybrid routing · Flow table constraint

1 Introduction

SDN is a new networking paradigm that decouples the control plane and data forwarding plane of network devices. More specifically, the controller constitutes the control plane of an SDN, and determines the forwarding path of each flow in a network with the centralized control manner. SDN switches constitute the data plane of an SDN, and response for data forwarding of each flow. Because the controller has global visibility and full control capacity over the whole network, SDN users can composite application programs run on top of the controller to monitor and manage the entire network (*e.g.*, traffic engineering, heavy hitter identification and proper routing) in an efficient and centralized manner. Thus, SDN has been used in different fields, such as campus networks and datacenter networks.

However, with the growth of Internet services, many large-scale data intensive applications (*e.g.*, video conferences, cloud services and financial data analysis) have become popular. Large-scale networks are experiencing more and more bursty flows. For example, in a practical datacenter network with 1500 server operational clusters, the average arrival rate will reach 10^5 flows per second for the core switches [6]. That means, if we perform per-flow routing (*i.e.*, one rule

© Springer International Publishing AG 2017

DOI: 10.1007/978-3-319-60033-8_9

for one flow) in that situation, it may require tens of thousands flow rules on each switch. However, due to the high price and energy-consuming of Ternary Content Addressable Memory (TCAM), an SDN switch usually contains a few thousand of flow rules [3]. Besides, recent testing results show that it usually takes 3.3 ms delay to insert a single flow rule into the flow table on a commodity switch [5]. On the other hand, deployment delay is critical for many applications, *e.g.*, the authors of [12] showed that 100 ms delay causes a 1% drop in revenue at Amazon and 400 ms delay causes a 5–9% decrease in traffic at Google. Thus, considering the flow table and deployment delay constraints, *per-flow routing is impractical to large-scale networks.*

To solve the flow table and deployment delay constraints problem, the classical design principle is destination host-based aggregate routing (*i.e.*, wildcard routing). However, now many datacenter networks contain millions of virtual or physical hosts, this scheme also requires tens of thousands TCAM rules to deploy host-based wildcard routing in large-scale networks [7]. Thus, the deployment delay and table size constraints in large-scale networks cannot be solved by destination host-based aggregate routing, especially for core switches, which may encounter a vast number of flows. Besides, the network performance (*e.g.*, load balancing) cannot be guaranteed by aggregate routing.

Thus this paper first studies wildcard routing and proposes the tag-based wildcard routing (TRPS). Then, we focus on the hybrid routing by joint optimization of per-flow routing and tag-based routing (HR-JPT). To the best of our knowledge, our work is the first to deploy hybrid routing by joint optimization of per-flow routing and tag-based routing under flow table size and deployment delay constraints for load balancing.

2 Tag-Based Rule Placement Scheme (TRPS)

A typical datacenter network architecture usually consists of three-level trees of switches. Typically, a three-tiered design can support tens of thousands of terminals. Under this circumstance, the bottleneck of the datacenter networks usually is the core switches, which encounter a massive number of flows [2]. Besides, due to the development of Open vSwitch and Overlay technologies, the resource shortages of edge switches have been eased [11]. Thus, our proposed scheme mainly concentrates on solving the deployment delay and flow table constraints of the core switches (or internal switches).

An openflow-based flow rule mainly contains match fields, priority, counters and actions. Match fields match against packet headers and consist of the *in_port*, *vlan_id*, *eth_dst*, *eth_src*, *ip_dst*, *ip_src etc.* Note that, with the update of the openflow protocol, the number of supported match fields are increasing and have some reserved match fields. Thus, we can define a *SwID* field (choose one reserved match field) as a new match field to perform the tag-based rule placement scheme (TRPS). The work [2] showed that the operation of adding or deleting a tag is easy for the controller, so the tag-based method is practical.

We illustrate TRPS by an example, as shown in Fig. 1. The controller first assigns four unique *SwID*: v_1, v_2, v_3, v_4 to corresponding switches. Next, The

Flow Table (v_1)

Dst	SwID	Action
*	SwID = v_2	Output=2
*	SwID = v_3	Output=3
*	SwID = v_4	Output=2
Dst = 3.0.0.1		Output=3
Dst = 2.0.0.1		SwID=v_4

Flow Table (v_4)

Dst	SwID	Action
*	SwID = v_1	Output=4
*	SwID = v_2	Output=4
*	SwID = v_3	Output=3
Dst = 3.0.0.1		Output=2
Dst = 2.0.0.1		Del SwID Output=1

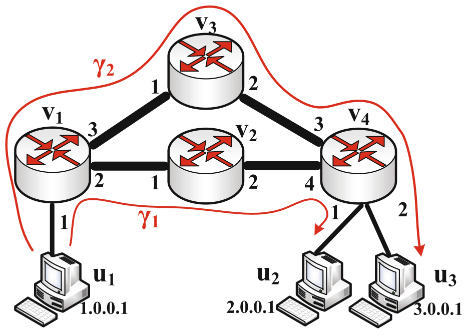

Fig. 1. Illustration of tag-based rule placement scheme. The left two plots denote the flow table of switch v_1 and v_4, respectively. The $SwID$ rules are pre-deployed when the topology is created. The Dst rules are reactively deployed based on the controller commands. When v_1 receives packets whose destination is u_2 with IP address 2.0.0.1, this switch will set $SwID = v_4$ for this flow and forward this flow based on switch level. When v_1 receives packets whose destination is u_3 with IP address 3.0.0.1, this switch will forward this flow based on per-flow level.

controller installs the tag-based switch level routing rules into switches. We only list dst match field and $SwID$ match field in Fig. 1 for simplicity. Due to the limited space, we only analyze switch v_1 to illustrate switch level rules. V_1 installs three switch level rules for other three switches, respectively. For example, the rule "$SwID = v_4, output = 2$" denotes that the flows labeled $SwID = v_4$ will be forwarded to port 2. Note that, all the flows labeled this $SwID$ have the same egress switch v_4. After all the proactive rules are installed, there will have three proactive rules on each switch. There arrive two flows: one is from 1.0.0.1 to 2.0.0.1 (denoted by γ_1) and the other is from 1.0.0.1 to 3.0.0.1 (denoted by γ_2). We assume that the controller decides to let γ_1 forwarded by tag-based routing and let γ_2 forwarded by per-flow routing. Thus, the controller only needs to interact with the ingress switch v_1 and egress switch v_4 to install tag-based routing rules. For the ingress switch v_1, we install a rule that matches flows with destination 2.0.0.1, the action of this rule is to attach $SwID = v_4$ so that they can be forwarded by tag-based routing rules. For the egress switch v_4, we install a rule that forwarded flows to the port 1, which matches flows with destination 2.0.0.1, so that these flows can reach the destination terminal. For γ_2, the controller needs interact with v_1, v_3, v_4 to install three rules for γ_2, which is same as the traditional per-flow routing rules.

By building rules on switch level, TRPS has the following advantages compared with other schemes:

(1) Reduce the numb er of required flow rules. The number of switches is always much smaller than the number of hosts in a network. For example, there have n switches and m hosts ($m \gg n$), if we build rules on host level, for the core switch, there need $m \times (m-1)$ rules at most. Even we build destination host-based rules, the core switch also needs m rules. But using TRPS, there need $n-1$ rules at most for the core switches.

(2) Decrease the deployment delay. TRPS combines the proactive routing and reactive routing. Since the switch level routing has been deployed in advance, the controller only needs to install 2 rules on the ingress and egress switches for each request. Let ψ denote the average length of all the paths, obviously, ψ increases along with the growing network and is usually greater than 5. The traditional scheme needs install ψ rules on average for each request. That means, our proposed wildcard routing scheme is efficient to reduce the deployment delay.

(3) Relieve the load of the controller. For the same reason, the controller only needs to send flow-mod commands to the ingress and egress switches with TRPS. That means this proposed scheme can relieve the load of the controller.

Comparing with other wildcard routing scheme, TRPS can achieve smaller deployment delay and similar network performance while using fewer flow rules.

3 Definition of Hybrid Routing (HR-JPT)

Although TRPS can save TCAM resources, reduce the deployment delay and relieve the load of the controller compared with other wildcard routing schemes, **if we deploy tag-based routing rules for all the flows, the network performance may become worse comparing with per-flow routing scheme.** Thus, in this section, we study the problem of *hybrid routing by joint optimization of per-flow routing and tag-based routing (HR-JPT)* to make use of the limited flow rules.

3.1 Network Model

An SDN typically consists of three device sets: a terminal set, $U = \{u_1, ..., u_m\}$, with $m = |U|$; an SDN switch set, $V = \{v_1, ..., v_n\}$, with $n = |V|$; and a cluster of controllers. The controllers response for route selection of all the flows and not participate in the packet forwarding in a network. These switches and terminals comprise the data plane of an SDN. Thus, in the view of the data plane, the network topology can be modeled by a directed graph $G = (U \cup V, E)$, where E is the link set in the network. For ease of expression, let $c(e)$ and $T(v)$ denote the capacity of link $e \in E$ and the number of available rules of switch $v \in V$ in graph G, respectively.

3.2 Hybrid Routing by Joint Optimization of Per-Flow Routing and Tag-Based Routing (HR-JPT)

There arrives a set of bursty flows, denoted by $\Gamma = \{\gamma_1, ..., \gamma_{|\Gamma|}\}$, in the network. By collecting the flow statistics information from switches, the controller can estimate the size (or intensity) of each flow $\gamma \in \Gamma$ as $f(\gamma)$. We also assume that each flow is unsplittable for simplicity. We use $\mathcal{P}_\gamma$ represents several feasible paths from source to destination for each flow $\gamma \in \Gamma$. Flow γ can be forwarded by a tag-based path and the tag-based path $h(\gamma) \in \mathcal{P}_\gamma$. One tag-based path can be shared by many flows and should be classed as default path (wildcard path). Note that, the $h(\gamma)$ is pre-deployed on switch level (may be the shortest path from ingress switch to egress switch) as illustrated in Sect. 2. Let $\mathcal{P}'_\gamma = \mathcal{P}_\gamma - h(\gamma)$ denote the feasible per-flow paths set of flow γ. Besides, let p_{ie} denote the set of ingress and egress switches of path $p \in \mathcal{P}_\gamma$.

We formulate the HR-JPT problem into a non-linear program as follows. Let variable $y_\gamma^p \in \{0, 1\}$ denote whether the flow γ selects the feasible path $p \in \mathcal{P}_\gamma$ or not. z_v^u denotes whether a tag-based flow rule need to be installed for the terminal u on switch v or not. H is the union of all the $h(\gamma)$. Due to the deployment delay is linearly associated with the number of rules that will be installed on each switch, we can use converting factor $\omega(v)$ to combine the two constraints as one constraint for simplicity. For example, the number of available rules $T(v)$ on switch v is 2 K and we want to be able to forward these flows within 3.3 s (denoted by T_0). It takes 3.3 ms (denoted by t_0) to insert a rule [5]. Considering the flow table constraint, we can only install 2 K rules at most. Considering the deployment delay constraint, we can only install 3.3 s ÷ 3.3 ms = 1 K rules at most. So, let $\omega(v) = 0.5$ to satisfy both constraints. In other words, we set:

$$\omega(v) = \min\{\frac{T_0}{t_0 \cdot T(v)}, 1\}, \quad \forall v \in V \tag{1}$$

HR-JPT solves the following problem:

$$\min \quad \lambda$$

$$S.t. \begin{cases} \sum_{p\in\mathcal{P}_\gamma} y_\gamma^p = 1, & \forall \gamma \in \Gamma \\ y_\gamma^p \le z_v^{d(p)}, & \forall v \in p_{ie}, p \in H \\ \sum_{\gamma\in\Gamma} \sum_{v\in p: p\in\mathcal{P}'_\gamma} y_\gamma^p + \sum_{u\in U} z_v^u \le \omega(v)\cdot T(v), & \forall v \in V \\ \sum_{\gamma\in\Gamma} \sum_{e\in p: p\in\mathcal{P}_\gamma} y_\gamma^p f(\gamma) \le \lambda \cdot c(e), & \forall e \in E \\ y_\gamma^p \in \{0,1\}, & \forall p, \gamma \\ z_v^u \in \{0,1\}, & \forall u \in U, v \in V \end{cases} \tag{2}$$

The first set of equations means that each flow will be assigned a feasible path from the source to the destination. The second set of inequalities denotes that, the tag-based rules will be deployed if at least one flow chooses this tag-based path as its route. As illustrated in Sect. 2, we only need to install rules on the ingress and the egress switches. The third set of inequalities denotes the flow table constraint and deployment delay constraint. The fourth set of inequalities expresses that the traffic load on each link e does not exceed the $\lambda \cdot c(e)$, where λ is the load balancing factor. Our objective is to minimize the load balancing factor, λ.

Theorem 1. *The HR-JPT problem defined in Eq.* (2) *is an NP-hard problem.*

Proof. We consider a special example of the HR-JPT problem, in which there is no flow table constraint and deployment delay constraint. Then, we are able to deploy routes of all the flows in an SDN so as to achieve the load balancing under link capacity constraint. In other words, this becomes an unsplittable multi-commodity flow with minimum congestion problem [4], which is NP-Hard. Since the multi-commodity flow problem is a special case of our problem, the HR-JPT problem is NP-Hard too.

4 Algorithm Description for the HR-JPT Problem

4.1 Approximation Algorithm to Solve HR-JPT

Due to NP-hardness, this section presents an approximation algorithm to deal with the HR-JPT problem. We design the Rounding-based Route Joint Deployment (RRJD) algorithm to solve the HR-JPT problem which is NP-hard. To solve this problem in polynomial time, we relax this assumption to suppose that each flow can be split and forwarded through multiple paths. By relaxing this assumption, y_γ^p and z_v^u are fractional. So we can solve it in polynomial time with a linear program solver (*e.g.*, CPLEX). Assume that the optimal solution is denoted by $\widetilde{y}$, and the optimal result is denoted by $\widetilde{\lambda}$. As the linear program is a relaxation of the HR-JPT problem, $\widetilde{\lambda}$ is a lower-bound result for this problem. More specifically, for each flow $\gamma \in \Gamma$, we select a feasible path $p \in P_\gamma$ with the probability of $\widetilde{y}_\gamma^p$ for flow γ. If $\exists p \in P_\gamma, \widehat{y}_\gamma^p = 1$, this means that flow γ selects $p \in P_\gamma$ as its finally route path. For the tag-based routing, $\widehat{x}_v^u = \max\{\widehat{y}_\gamma^p, \forall d(p) = u, v \in p_{ie}, p \in H, \gamma \in \Gamma\}$. By this way, we have determined the final route paths for all the flows. The RRJD algorithm is formally described in Algorithm 1.

Algorithm 1. RRJD: Rounding-based Route Joint Deployment

```
1: Step 1: Solving the Relaxed HR-JPT Problem
2: Construct a linear program $LP_1$ based on Eq. (2)
3: Obtain the optional solution $\widetilde{y}$
4: Step 2: Route Selection for Load Balancing
5: Derive an integer solution $\widehat{y}_\gamma^p$ by randomized rounding
6: for each switch $v \in V$, each terminal $u \in U$ do
7:     $\widehat{x}_v^u = \max\{\widehat{y}_\gamma^p, \forall d(p) = u, v \in p_{ie}, p \in H, \gamma \in \Gamma\}$
8: for each flow $\gamma \in \Gamma$ do
9:     for each route path $p \in P_\gamma$ do
10:        if $\widehat{y}_\gamma^p = 1$ then
11:            Appoint path $p$ for flow $\gamma$
```

4.2 Approximate Performance Analysis

We give two famous lemmas for probability analysis.

Theorem 2 (Chernoff Bound). *Given n independent variables: $x_1, x_2, ..., x_n$, where $\forall x_i \in [0,1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^{n} x_i]$. Then,* $\mathbf{Pr}\left[\sum_{i=1}^{n} x_i \geq (1+\epsilon)\mu\right] \leq e^{\frac{-\epsilon^2\mu}{2+\epsilon}}$, *where ϵ is an arbitrarily positive value.*

Theorem 3 (Union Bound). *Given a countable set of n events: $A_1, A_2, ...,$ A_n, each event A_i happens with possibility $\mathbf{Pr}(A_i)$. Then, $\mathbf{Pr}(A_1 \cup A_2 \cup ... \cup A_n) \leq \sum_{i=1}^{n} \mathbf{Pr}(A_i)$.*

We define a variable α as follows:

$$\alpha = \min\{\min\{\frac{\widetilde{\lambda} c_{\min}}{f(\gamma)}, \gamma \in \Gamma\}, \min\{T(v), v \in V\}\} \tag{3}$$

Link Capacity Constraint. The load of link e from each flow γ is defined as a variable $x_{e,\gamma}$. we have:

$$\mathbb{E}\left[\sum_{\gamma \in \Gamma} x_{e,\gamma}\right] = \sum_{\gamma \in \Gamma} \mathbb{E}\left[x_{e,\gamma}\right] = \sum_{\gamma \in \Gamma} \sum_{e \in p: p \in \mathcal{P}_\gamma} \widetilde{y}_\gamma^p f(\gamma) \leq \widetilde{\lambda} c(e) \tag{4}$$

Combining Eq. (4) and the definition of α in Eq. (3), we have

$$\begin{cases} \frac{x_{e,\gamma} \cdot \alpha}{\widetilde{\lambda} c(e)} \in [0,1] \\ \mathbb{E}\left[\sum_{\gamma \in \Gamma} \frac{x_{e,\gamma} \cdot \alpha}{\widetilde{\lambda} \cdot c(e)}\right] \leq \alpha. \end{cases} \tag{5}$$

By applying Theorem 2, we assume that ρ is an arbitrary positive value. It follows

$$\mathbf{Pr}\left[\sum_{\gamma \in \Gamma} \frac{x_{e,\gamma} \cdot \alpha}{\widetilde{\lambda} \cdot c(e)} \geq (1+\rho)\alpha\right] \leq e^{\frac{-\rho^2\alpha}{2+\rho}} \tag{6}$$

Now, we assume that

$$\mathbf{Pr}\left[\sum_{\gamma \in \Gamma} \frac{x_{e,\gamma}}{\widetilde{\lambda} \cdot c(e)} \geq (1+\rho)\right] \leq e^{\frac{-\rho^2\alpha}{2+\rho}} \leq \frac{\mathcal{F}}{n^2} \tag{7}$$

where $\mathcal{F}$ is the function of network-related variables (such as the number of switches n, *etc.*) and $\mathcal{F} \to 0$ when the network size grows.

The solution of Eq. (7) is expressed as:

$$\rho \geq \frac{\log \frac{n^2}{\mathcal{F}} + \sqrt{\log^2 \frac{n^2}{\mathcal{F}} + 8\alpha \log \frac{n^2}{\mathcal{F}}}}{2\alpha}, \quad n \geq 2 \tag{8}$$

Lemma 1. *The proposed RRJD algorithm achieves the approximation factor of* $\frac{4\log n}{\alpha}+3$ *for link capacity constraint.*

Proof. Set $\mathcal{F}=\frac{1}{n^2}$. Equation (7) is transformed into:

$$\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{x_{e,\gamma}}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho)\right]\leq\frac{1}{n^4},\quad \text{where}\quad \rho\geq\frac{4\log n}{\alpha}+2 \tag{9}$$

By applying Lemma 3, we have,

$$\mathbf{Pr}\left[\bigvee_{e\in E}\sum_{\gamma\in\Gamma}\frac{x_{e,\gamma}}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho)\right]\leq n^2\cdot\frac{1}{n^4}=\frac{1}{n^2},\quad \rho\geq\frac{4\log n}{\alpha}+2 \tag{10}$$

The approximation factor of our algorithm is $\rho+1=\frac{4\log n}{\alpha}+3$.

Flow Table Constraint. Note that, the approximate performance analysis of the **deployment delay constraint** is same as the analysis of the flow table constraint, we omit it due to limited space. Similar to Link Capacity Constraint, we define random variable δ, $t_{v,\gamma}$ and $\mathcal{F}$. we have:

$$\delta\geq\frac{\log\frac{n}{\mathcal{F}}+\sqrt{\log^2\frac{n}{\mathcal{F}}+8\alpha\log\frac{n}{\mathcal{F}}}}{2\alpha},\quad n\geq 2 \tag{11}$$

We give the approximation performance as follows.

Lemma 2. *After the rounding process, the total number of flow rules on any switch* v *will not exceed the constraint* $T(v)$ *by a factor of* $\frac{3\log n}{\alpha}+3$.

Proof. Set $\mathcal{F}=\frac{1}{n^2}$. $\mathcal{F}\to 0$ when $n\to\infty$. With respect to Eq. (11), we set

$$\delta=\frac{\log\frac{n}{\mathcal{F}}+\log\frac{n}{\mathcal{F}}+4\cdot\alpha}{2\cdot\alpha}=\frac{6\log n+4\cdot\alpha}{2\cdot\alpha}=\frac{3\log n}{\alpha}+2 \tag{12}$$

Then Eq. (12) guarantees with $1+\delta=\frac{3\log n}{\alpha}+3$, which concludes the proof.

Approximation Ratio. With these analyses, we know the approximation factor for link capacity constraint and flow table constraint (deployment delay constraint) is $\frac{4\log n}{\alpha}+3$ and $\frac{3\log n}{\alpha}+3$, respectively. More specifically, by using the proposed RRJD approximate algorithm, we can scale flows by a factor of $\frac{4\log n}{\alpha}+3$ to satisfy the link capacity constraint and by a factor of $\frac{3\log n}{\alpha}+3$ to satisfy the flow table constraint (deployment delay constraint). For example, let $\alpha=100$, n = 100, then the approximation factor for link capacity constraint and flow table constraint (deployment delay constraint) is 3.08 and 3.06, respectively.

5 Simulation Results

5.1 Performance Metrics and Setting

The proposed scheme and algorithm focus on datacenter networks, so we choose the fat-tree topology [7] in the simulation, which has been widely used in many datacenter networks. The fat-tree topology has 16 core switches, 32 aggregation switches, 32 edge switches and 128 servers. We also assume that each server performs 15 VMs (Virtual Machines) for simulating the realistic scenario. Thus, the total number of terminals is 1920 and we set all links are 1 Gbps for simplicity. Each simulation will be executed 100 times and average the numerical results. We use the power law for the flow-size distribution, where 20% of all flows account for 80% of traffic volume.

Since this paper studies both the wildcard routing and the hybrid routing, we let TRPS denote all the flows are forwarded by tag-based routing. The RRJD denotes the hybrid routing by joint per-flow routing and tag-based routing. We compare these two proposed methods with four other methods: (1) The first one is the per-flow routing (**PFR**). We adopt the multicommodity flow (MCF) method using randomized rounding for unsplittable flows in an SDN. Note that, the method may drop some flows to satisfy the flow table constraint and deployment delay constraint. (2) The second one is **OSPF** protocol, which is the host-based routing [10]. This benchmark is mainly compared with TRPS (both are wildcard routing scheme). (3) The third one is DomainFlow [9] (denoted by **DFW**). DFW divided the network into two parts, one part using OSPF (wildcard rules) and another part using per-flow routing rules. This benchmark is mainly compared with RRJD (both are hybrid routing scheme). (4) The last one is the optimal result for the linear program LP_1 based on Eq. (2), denoted by **OPT**. Since LP_1 is the relaxed version of the HR-JPT problem, OPT is a lower-bound for HR-JPT.

We mainly adopt 5 different metrics for performance measurement. The first two metrics are the maximum/average number of required flow rules on all the switches. After executing these algorithms, we can obtain the number of used flow rules on each switch, and compute the maximum/average number of used flow rules on all the switches. To measure the network performance, the next two metrics are link load ratio (LLR) and network throughput (NT). LLR can be obtained by measuring the traffic load $l(e)$ of each link e. Then, LLR is defined as: $LLR = \max\{l(e)/c(e), e \in E\}$. The smaller LLR means better load balancing. NT can be obtained by measuring the total traffic amount through the network. The last metric is the communication overhead to/from the controller, which can reflect the overhead of the controller.

5.2 Simulation Evaluation

We run 6 sets of experiments to test the 5 different metrics of these algorithms. The flow table constraint is 5 k [3]. The first two sets of experiments observe the maximum/average number of required flow rules by increasing the number of flows. The results are showed in Figs. 2 and 3. Due to the flow table constraint, the maximum number of flow rules is 5 K. Both figures show that the wildcard routing (*i.e.*, OSPF and TRPS) require fewer rules than other methods. Besides, our proposed TRPS wildcard routing can reduce the maximum/average number of required rules by about 20%/65%. Note that, due to TRPS can significantly reduce the number of rules for the core switches and have less impact on the edge switches, the performance of the average number of required rules is much better. These two figures indicate that TRPS needs fewer rules than OSPF.

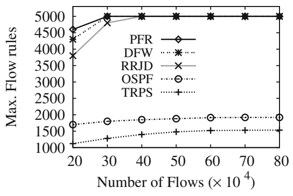

Fig. 2. Max. num. of flow rules vs. num. of flows

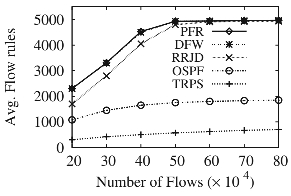

Fig. 3. Avg. num. of flow rules vs. num. of flows

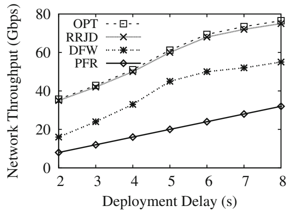

Fig. 4. Network throughput vs. deployment delay

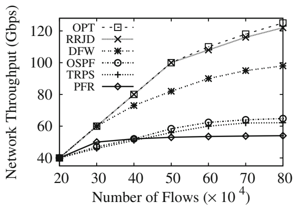

Fig. 5. Network throughput vs. num. of flows

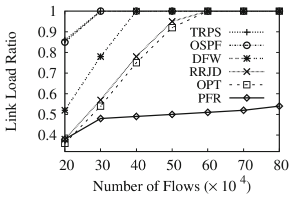

Fig. 6. Link load ratio vs. num. of flows

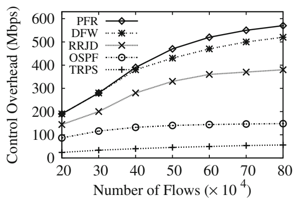

Fig. 7. Communication overhead vs. num. of flows

The third set of experiments observes the network throughput by changing the deployment delay constraint. The default number of flows is 40 W. The result in Fig. 4 indicates that RRJD can improve network throughput by about 43% compared with DFW. The performance of PFR is worse due to the low speed of per-flow routing.

As illustrated in Fig. 5, the fourth set of experiments measures the change of the network throughput by increasing the number of flows. Our proposed RRJD can improve the network throughput by about 30%/90% compared with DFW/PFR while using a similar number of rules. Besides, the network throughput of OSPF and TRPS are similar while our proposed TRPS can reduce the required number of rules by about 65% on average (illustrated by Fig. 3). The worse performance of PFR indicates that per-flow routing is impractical for large-scale networks.

The result of the fifth set is presented in Fig. 6. PFR has exceptionally low link load ratio because of the flow table constraint and many flows are dropped. Our proposed RRJD can reduce link load ratio by about 23% compared with DFW while using a similar number of rules. Besides, all the above three figures show that the performance of OPT and RRJD is very similar, that means, the proposed algorithm can elegantly solve the HR-JPT problem and get a feasible solution that closing to the lower-bound OPT.

The last set of simulations evaluates the communication overhead between the controller and switches. The wildcard routing (OSPF and TRPS) can greatly reduce the control overhead, because many flows can be forwarded by wildcard rules without being reported to the controller. Our proposed TRPS can reduce overhead by about 75% compared with OSPF, this is because of switch-based rules and proactive rules. Our proposed RRJD can reduce overhead by about 30% compared with DFW, this is also due to the use of proactive rules and switch-base rules (Fig. 7).

From these simulations, we can make some conclusions. (1) Our proposed TRPS (the version of only tag-based wildcard routing) is better than OSPF (terminal-based wildcard routing). For example, TRPS can reduce the number of required rules by about 65% on average and reduce the overhead by about 75% compared with OSPF. (2) Our proposed RRJD increases network throughput by about 43% (or 30%) compared with DFW by changing the number of rules (or changing the deployment delay). Moreover, RRJD can reduce the communication overhead by about 30% and reduce link load ratio by about 23% compared with DFW. (3) The performance of our proposed RRJD is similar with the lower-bound OPT, which means the approximation algorithm is efficient to solve the NP-Hard problem.

6 Related Works

Since routing is a critical issue to achieve better network performance in an SDN, there are many related works to handle the routing problem. The obvious way is to deploy one individual rule for each flow to provide the fine-grained route selection. Al-Fares *et al.* [1] designed a dynamic flow scheduling for datacenter networks that sets up a new TCAM rule for every new flow in the network. However, as the networks are experiencing more and more flows while the commodity switches only contain a few thousand TCAM rules [3], per-flow routing is impractical to large-scale networks.

Some works devoted to aggregate traffic (*i.e.*, wildcard routing). iSTAMP [8] use part of the TCAM rules for aggregation traffic. But it may encounter the aggregation feasibility problem in a network for these works. Some works considered the wildcard routing. Work [10] adopted the destination terminal-based aggregate routing. However, now many datacenter networks contain millions of virtual or physical end terminals, this scheme also requires tens of thousands TCAM rules to deploy terminal-based wildcard routing in large-scale networks [7]. Moreover, these works suffered from worse network performance due to that many flows will be forwarded by the same path, which might cause link congestion.

To achieve the balance between network performance and flow table constraint, some works studied the combining of per-flow routing and wildcard routing. Devoflow [3] combined pre-deployed wildcard rules and dynamically-established exact rules, DomainFlow [9] divided the network into two parts, one part using wildcard rules and another part using exactly matching rules. However, these works did not mention the details of how to deploy default paths and mainly adopted the OSPF protocol as the wildcard routing. The performance of OSPF protocol cannot be guaranteed.

7 Conclusion

In this paper, we proposed a novel tag-based rule placement scheme (TRPS) for wildcard routing and studied the hybrid routing problem that joint optimization of per-flow routing and tag-based routing (HR-JPT). The testing results have shown high efficiency of TRPS (compared with other wildcard routing schemes, *e.g.*, OSPF) and HR-JPT (compared with other hybrid routing schemes, *e.g.*, DFW). In the future, we will consider the update scheme and online algorithm.

Acknowledgments. This paper is supported by the NSFC under No. 61472385 and U1301256, and NSF of Jiangsu in China under No. BK20161257.

References

1. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.: Hedera: dynamic flow scheduling for data center networks. In: NSDI, vol. 10, p. 19 (2010)
2. Banerjee, S., Kannan, K.: Tag-in-tag: efficient flow table management in SDN switches. In: International Conference on Network and Service Management, pp. 109–117 (2014)
3. Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S.: Devoflow: scaling flow management for high-performance networks. In: ACM SIGCOMM Computer Communication Review, vol. 41, pp. 254–265. ACM (2011)
4. Even, S., Itai, A., Shamir, A.: On the complexity of time table and multi-commodity flow problems. In: 1975 16th Annual Symposium on Foundations of Computer Science, pp. 184–193. IEEE (1975)
5. Huang, H., Guo, S., Li, P., Ye, B., Stojmenovic, I.: Joint optimization of rule placement and traffic engineering for QoS provisioning in software defined network. IEEE Trans. Comput. **64**(12), 3488–3499 (2015)
6. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: measurements and analysis. In: ACM SIGCOMM Conference on Internet Measurement 2009, Chicago, Illinois, USA, November, pp. 202–208 (2009)
7. Lu, X., Xu, Y.: Sfabric: a scalable SDN based large layer 2 data center network fabric. In: 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), pp. 57–58. IEEE (2015)
8. Malboubi, M., Wang, L., Chuah, C.N., Sharma, P.: Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp). In: IEEE 2014 Proceedings of the INFOCOM, pp. 934–942. IEEE (2014)

9. Nakagawa, Y., Hyoudou, K., Lee, C., Kobayashi, S., Shiraki, O., Shimizu, T.: Domainflow: practical flow management method using multiple flow tables in commodity switches. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, pp. 399–404. ACM (2013)
10. Network, M.Y.O.: OSPF Network Design Solutions (2003)
11. Wang, A., Guo, Y., Hao, F., Lakshman, T., Chen, S.: Scotch: elastically scaling up SDN control-plane using vswitch based overlay. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, pp. 403–414. ACM (2014)
12. Wei, C., Buffone, R., Stata, R.: System and method for website performance optimization and internet traffic processing. US Patent 8,112,471, 7 Feb 2012