Routing Optimization For Cloud Services in SDN-based Internet of Things With TCAM Capacity Constraint

Xiong Wang Guangxu Yang Jing Ren Sheng Wang Shizhong Xu
Key Lab of Optical Fiber Sensing & Communications
School of Information & Communication Engineering
University of Electronic Science & Technology of China, Chengdu, China
{wangxiong, renjing, wsh_keylab, xsz}@uestc.edu.cn, yanggx187@gmail.com

Abstract*—Distributed in-network cloud architecture is a promising solution to efficiently host next generation Internet-of-Things (IoT) services. With the rapid increase of IoT devices and applications, the backhaul or backbone networks, which transmit IoT traffic to various in-network clouds, will experience a predicted explosion in the volume of carried traffic. To guarantee the QoS of IoT cloud services and improve the network performance, it is crucial for network operator to implement efficient routing optimization strategies for IoT traffic. As a promising networking paradigm, Software-Defined Networking (SDN) has flexible and programmable control capability for fine-grained flows. The emergence of SDN paves a way for implementing high-performance routing optimization in networks. In SDN networks, the routing strategies are realized through flow rules, which are usually stored in TCAM with very limited capacity. However, the number of IoT flows are enormous. Thus, in this paper, we address the routing optimization problem in SDN-based IoT with TCAM capacity constraint. We first formulate the problem as a mixed integer linear programming problem and prove the problem is NP-hard. Then to solve the problem efficiently, we propose several approximate algorithms, which solve the problem in two stages. In the first stage, the algorithms calculate the routing strategies for flows without considering the TCAM capacity constraint. To meet the TCAM capacity constraint, the algorithms using different strategies to adjust the paths of some flows in the second stage. Extensive simulations are conducted on both real ISP and synthetic topologies to evaluate the performance of the algorithms. The simulation results verify that the algorithms can achieve promising load balancing performance in SDN-based IoT, where the capacity of TCAM in SDN switches is very limited.

I. Introduction

Internet of Things (IoT) is an emerging network paradigm, which refers to the networking of devices, vehicles, buildings, machines, and other objects with embedded sensing, computing, and communication capabilities. The connected IoT objects sense, process and share real-time information about the physical world [1]. The IoT makes the Internet even more pervasive, and by enabling easy access and interaction with various devices, the IoT will boost the development of applications that make use of the huge amount and variety of data generated by such objects to provide new services for our daily life. This emerging paradigm enables a new breed of applications in many different domains, such as medical aids, industrial automation, smart grids, intelligent home, and intelligent transportation [2].

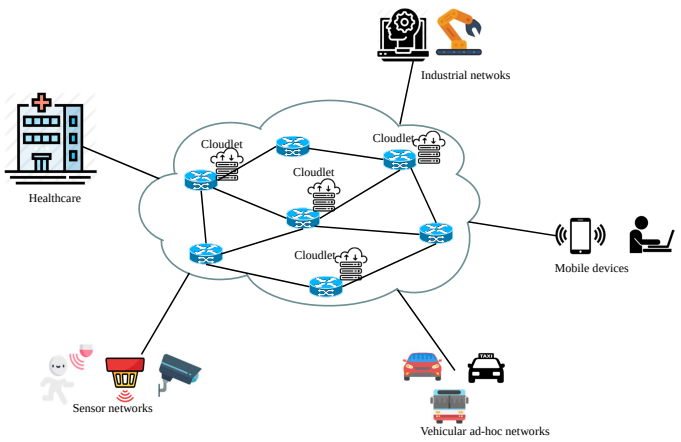


Fig. 1: The architecture of IoT

Due to the intrinsic limitations of lightweight IoT devices (e.g., battery capacity, computation power, and storage space), many of the applications process and analyse the vast amounts of data collected from multiple devices in remote data centers with extensive resource to create meaningful information for the end users. However, with the rapid increase of IoT applications and devices, moving vast amounts of data to a few large data centers with concentrated computing and storage resources will inevitably lead to excessive network load and high service latencies. Therefore, to meet the QoS requirements of IoT applications while improving the network resource utilization efficiency, the centralized large data centers will be probably replaced by some small data centers, which are distributively located at different network nodes in future networks (as shown in Fig. 1). These small data centers are usually referred to as cloudlets [3] or micro-clouds [4].

On the other hand, with the predicted explosion in the number of IoT applications and connected devices [5], the volume of data required to be sent to nearby clouds through backhaul network will increase explosively in the near future. In other words, the number of flows and the total volume of flows transmitted from IoT devices to Cloudlets will be huge, which may lead to high network congestion. Furthermore, many IoT applications, such as remote medical aids,

intelligent transportation, and industrial automation, have strict QoS requirements. Generally, there are two ways to avoid network congestion and guarantee the QoS requirements of IoT applications. The first way is to upgrade network devices to provide higher bandwidth. The another way is to implement Routing Optimization (RO) strategies for the flows generated by IoT applications.

The Routing Optimization (RO) is to adapt the routing of traffic to the changing traffic patterns. Since RO is an efficient way to improve user experience and network performance without upgrading network infrastructure, the RO has attracted enormous attention from both academic and industrial communities [6] in the past decades. However, due to the lack of flexible routing control capacity, it is hard for the traditional networks to achieve satisfactory performance with acceptable cost. For example, the networks running OSPF routing protocol restrict all flows to be forwarded along the shortest paths decided by the link weights, where the traffic distribution can be optimized by adjusting the link weights. However, although the shortest path routing paradigm is easy to implement, it is hard to get the optimal distribution of traffic through a setting of link weights. In order to improve the network performance, some explicit routing protocols, such as Multi-Protocol Label Switching (MPLS) [7] and segment routing [8], are deployed in networks. The explicit routing protocols can realize the optimal traffic distribution by routing flows on carefully selected paths. However, these explicit routing protocols achieve the promising performance at the costs of establishing and maintaining a lot of paths, e.g., establishing a large amount of LSPs using MPLS is time and labor consuming, and the source routing technique used in SR requires extra bandwidth.

To enhance the control plane flexibility of networks, the Software-Defined Networking (SDN) is proposed [9]. SDN is a programmable networking paradigm, which separates the control and data planes in a network. The control plane runs on logically centralized network controllers, and the data plane located in each SDN switch. Network controllers provide program interfaces for various network applications and control the behaviors of SDN switches using standardized protocols (e.g., OpenFlow [10]). This functional separation and the implementation of control plane functions on powerful centralized platforms bring various expected operational benefits, such as programmability, flexibility, and global view of the network state. With the help of programmable and centralized network controllers, SDN networks have fine-grained and flexible routing control capability, which is achieved by modifying the flow rules residing in flow tables of SDN switches. Thus, the deployment of SDN paves a way for implementing efficient and online RO according to the changing traffic pattern. It has been shown that in SDN-enabled production networks, the RO can achieve near-optimal performance in the aspects of throughput and link utilization [11], [12].

In SDN networks, different routing strategies are realized by installing different flow rules in each SDN switch. To forward packets in line-rate, the flow rules are usually stored in Ternary Content-Addressable Memory (TCAM), which can perform parallel wildcard matching at high speed. However, TCAM has high power consumption and large footprint, leading to high heat generation [13], and TCAM is also very expensive [14]. Therefore, the capacity of TCAM in commodity SDN switches is very limited, e.g., many SDN switches usually only have 10K to 40K TCAM entries [15]. In contrast, there may be a huge number of cloud service flows (we use flows for short in this paper) going through an SDN switch in IoT, e.g., the number of fine-grained flows going through an SDN switch can be up to 1000K [16]. Thus, it is impossible to use a dedicated flow rule to process the packets of each flow. Since the Static Random Access Memory (SRAM) have a larger capacity and is cheaper than TCAM, some studies [17] implement flow tables in SRAM. But SRAM based flow tables have much higher implementation complexity than TCAM based flow tables, and it is also very challenging for the SRAM based flow tables to realize line-rate wildcard lookup [18]. Furthermore, the flow rule updating procedure in SDN switches is quite slow (e.g., about 40 to 50 rule updates per second [19]), updating a large number of flow rules will take a long time, which makes the routing strategies lag behind traffic variation. Therefore, a more efficient way is to calculate routing strategies under the TCAM capacity constraint.

In another hand, as it happens on most emerging network architectures and protocols, upgrading traditional networks to SDN networks cannot be realized at once, especially for large networks. The reason is twofold: First, an upgrade of an entire network requires huge capital expenditures since high-speed network device is expensive; Second, the one-step upgrade also raises performance and security risks due to the lack of operational experience for SDN networks. Therefore, large network providers usually prefer to incrementally deploy SDN devices in their existing networks. As a result, hybrid SDN architecture is likely to be a long-term solution for productive networks [20]. We also consider hybrid SDN networks in this paper.

The RO problem in SDN networks has attracted many research interests in recent years. However, to the best of our knowledge, the TCAM capacity constraint is not considered in the existing studies, which makes the existing algorithms hard to be used in networks carrying IoT traffic. In this paper, we investigate the RO problem under the TCAM capacity constraint in hybrid SDN-based IoT. Since full SDN-based IoT can be viewed as special cases of hybrid SDN-based IoT, the algorithms designed for hybrid SDN-based IoT in this paper can also be used in full SDN-based IoT. Our contributions are summarized as follows.

i) We address the RO problem under the TCAM capacity constraint in hybrid SDN-based IoT. We first formulate the problem as a Mixed Integer Linear Programming (MILP) problem, and then we prove that the problem is NP-hard.

ii) To efficiently tackle the problem, we propose several approximate algorithms with different complexity to solve the problem.

iii) We conduct extensive simulations to evaluate the performance of the proposed approximate algorithms. The simulation results show that the proposed approximate algorithms can find near-optimal routing solutions for the flows under the TCAM capacity constraints, and by carefully selecting

the paths for flows, we can achieve promising load balancing performance with a small number of TCAM entries in each SDN switch.

The remainder of this paper is organized as follows: Section II reviews related work. Section III formulates the RO problem in IoT. Section IV presents the approximate algorithms designed for solving the RO in IoT. Section V evaluates the proposed algorithms. Finally, Section VI summarizes the paper and presents the main conclusions.

II. Related Work

The RO aims to adapt the routing of traffic to network conditions, with the goals of improving user experience and network performance. Since RO is an efficient scheme to improve network service capability without upgrading network infrastructure, it has attracted extensive attentions from both academic and industrial communities. During the last decades, IP based [22], MPLS based [23], [24], and segment routing based RO [25] schemes have been proposed to improve the network performance and meet the Quality-of-Service (QoS) requirements of applications. We refer the reader to [26], [27] for exhaustive surveys of the work on the RO problem in traditional networks. However, due to the limitations of traditional network architecture, the existing RO schemes suffer from many problems, such as low flexibility, low scalability, low performance, and high operational cost [28].

To cope with the limitations of the traditional network architecture, the SDN paradigm is proposed. SDN decouples the forwarding and control planes of a network system [21], so that network operators can program packet forwarding behavior based on the global view of network status. The SDN pave a way for implementing flexible and fine-grained RO, which can significantly improve the network performance and resource utilization. Microsoft [11] and Google [12] have deployed SDN in their inter-data center networks, and their operational results show that the SDN-enabled networks can achieve near-optimal performance in the aspect of throughput and link utilization by implementing RO.

Most RO problems in full SDN networks (all nodes are SDN-enabled) are equivalent to the multicommodity flow problems [29], which have been well studied. Nevertheless, fully deploying the SDN in the existing network will not work out in a short term due to the budget and operation constraints [30], [31]. Deploying SDN in network incrementally should be a natural choice. Thus, the RO problem in hybrid SDN networks attracts more attentions [32]–[36]. S. Agarwal et al. [32] first address the RO problem in hybrid SDN networks. They formulate the RO problem as a linear programming problem and propose a polynomial algorithm with performance bound to calculate paths and flow splitting strategies for the flows. Similar to the work in [32], He et al. also propose polynomial algorithms with performance bound to solve the RO problems for two hybrid SDN network models. In [32], [33], the link weights are fixed and all link weights are assumed to be 1. To further improve the RO performance, Guo et al. [34] propose to jointly optimize OSPF link weights and flow splitting ratios of the SDN switches in hybrid SDN networks. Furthermore, to guarantee forwarding consistency in hybrid SDN network, Wang et al. [35] optimize traffic distribution on constructed forwarding graphs. In SDN networks, the SDN switches use flow rules usually installed in TCAMs to math and forward flows. Since the TCAM is expensive and power hungry, the capacity of TCAM is very limited. To mitigate the impact of TCAM space limitation on the RO performance, S. Zhang et. al [36] propose TCAM space aware RO algorithms, which try to minimize the TCAM space consumption while optimizing flow routing. Even though the TCAM space aware RO algorithms can reduce the required TCAM entries, they are also hard to meet the TCAM capacity constraint, especially when there are a large number of flows.

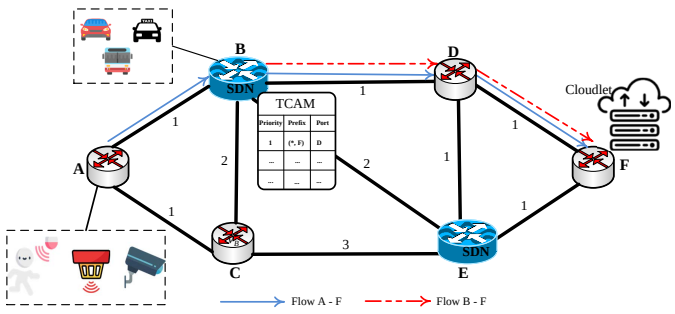
On the other hand, the SDN deployment strategy also has an impact on the RO performance of hybrid SDN networks. Y. Guo et al. [37] propose a heuristic algorithm to find a migration sequence of traditional routers to SDN switches that obtains the most of the benefits from the perspective of RO. Furthermore, K. Poularakis et. al [38] consider a more practical model that captures time-varying migration costs, network topologies, and two objectives benefiting for RO. These work consider the objectives benefiting for RO when make SDN deployment decisions. However, these work are different from the work in this paper for the following two reasons: First, these work pursue the indirect objectives benefiting for RO (e.g., maximize programmable traffic or routing flexibility), but our work directly optimizes the traffic distribution objective (e.g., load balancing); Second, these work calculate SDN deployment strategies, while our work optimizes the routing for flows.

In summary, the RO problem in hybrid SDN has gained many attentions due to the important role of RA. The SDN networks rely on flow rules installed in TCAM entries to realize routing strategies. So in real SDN networks, the routing strategies would likely be unrealizable due to the TCAM capacity limitation. Therefore, the TCAM capacity constraint must be considered when we optimize routing strategies, especially for IoT, where the number of flows is much more than that of available flow rules. However, to the best of our knowledge, all the existing work do not consider the TCAM capacity constraint.

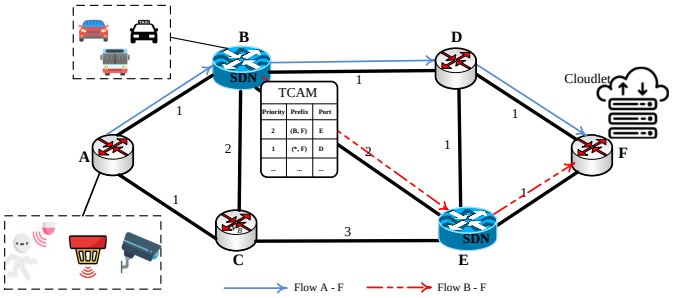
III. Problem Formulation

A. Network Model and Assumptions

We model an SDN-based IoT as a connected undirected graph $G(V, E)$, where V is the set of nodes and E is the set of links. The cloudlets [3], which provide computing and storage capabilities for IoT services, are deployed at some network nodes. Each link $e \in E$ has a capacity $c(e)$ and a weight $w(e)$. Since deploying SDN devices incrementally is a natural choice for network providers [38], we in this paper also consider hybrid SDN networks, where only a subset of the nodes are SDN switches and the rest of the nodes are traditional routers. We assume that the set of nodes deploying with SDN switches are given. Let V_{SDN} and V_{IP} ($V = V_{SDN} \cup V_{IP}$) denote the sets of SDN nodes and IP routers, respectively. The IP routers must forward packets to the adjacent nodes on the



(a)



(b)

Fig. 2: An example for routing optimization in hybrid SDN-based IoT

shortest paths determined by the link weights, while the SDN switches can forward packets to any adjacent nodes according to the rules stored in TCAM. Since TCAM is expensive and power hungry, the capacities of TCAM in switches is very limited. Let $\gamma(v)$ be the number of available (i.e., unused) TCAM entries in SDN node v ($v \in V_{SDN}$).

The set F of flows in the network is given. In SDN-based IoT, the granularity of flows can be flexibly defined by the matching fields allowed by SDN. For example, a flow can be either a TCP streaming video flow or an aggregated flow between two sub-networks. Each flow $f_i \in F$ can be represented as a three-tuple (s_i, d_i, h_i), where s_i and d_i are the source and destination nodes of f_i, respectively, and h_i is the volume of flow f_i. In hybrid SDN networks, some paths are infeasible due to the shortest path routing constraint of traditional IP routers. For example, the path $A-B-C-E-F$ in Fig. 2 is infeasible since IP router C running OSPF protocol cannot forward packets destined to node F via link (C, E). For hybrid SDN networks, a path p from source node s to destination node d is termed feasible if it satisfies the following two constraints:

c1) For each non-SDN node $u \in p$, link (u, v) is on the shortest path from node u to node d, where v is the next node of u on path p.

c2) The path p is loop-free.

We use P_i to denote the set of feasible paths for flow f_i, and the feasible path set P_i is calculated in advance. To reduce the complexity, we calculate at most k feasible paths for each flow. We use an algorithm modified from the k-shortest path algorithm to calculate at most k feasible paths for each flow. The k-shortest path algorithm generates candidate paths from each node on an existing path. However, in hybrid SDN networks, a non-SDN node cannot forward a flow to a neighbor that is not on the shortest path from the non-SDN node to the destination node of the flow. Thus, the modified algorithm only generates candidate paths from the SDN nodes on an existing feasible path. To ensure that the returned paths are feasible, the infeasible paths generated by the modified algorithm will be ignored. The algorithm for calculating feasible paths for a flow f_i is shown in **Aglorithm 1**.

Algorithm 1 Calculating Feasible Paths For a Flow

Input: Network topology $G(V, E)$ and a flow $f_i = (s_i, d_i, h_i)$.
Output: The set P_i of feasible paths for flow f_i.

$Q \leftarrow \emptyset$, $P_i \leftarrow \emptyset$
$sp_i \leftarrow$ the shortest paths from node s_i to d_i
$P_i \leftarrow P_i \cup sp_i$, $Q \leftarrow Q \cup sp_i$
while $|P_i| \leq k$ and Q is not empty **do**
  $p \leftarrow$ the shortest path in set Q
  $P_i \leftarrow P_i \cup p$
  $Q \leftarrow Q \backslash p$
  for each node $u \in p$ **do**
    if $u \in V_{SDN}$ and $u \neq d_i$ **then**
      $p_{s_i u} \leftarrow$ the path from s_i to u and included in path p
      for each node v adjacent to u and $v \notin p$ **do**
        $p_{v d_i} \leftarrow$ the shortest path from node v to node d_i
        $p \leftarrow p_{s_i u} \cup (u, v) \cup p_{v d_i}$
        $Q \leftarrow Q \cup p$
      end for
    end if
  end for
end while
return P_i

If there are enough TCAM entries in an SDN switch, the SDN switch can use a dedicated flow rule to forward each flow. However, it is not possible due to the hard constraint of TCAM capacity. Similar to the traditional IP routers, SDN switches can also aggregate the rules with the same prefix and action to one rule. To save TCAM entries, we assume that initially, all SDN switches use only one rule to forward the packets destined to each node, i.e., the rules for the packets with the same destination IP address are aggregated into one rule. Without loss of generality, we assume that in default, the flows are forwarded by aggregated rules, which forward flows along the shortest paths. To improve the RO objective, the available TCAM entries in SDN switches can be used to adjust the paths of the flows going through the SDN switches. Let us consider the example in Fig. 2. In Fig. 2, the numbers on the links denote the link weights, and there are two flows (f_{AF} and f_{BF}) requiring 1 unit bandwidth. Since the two flows are destined to the same node, the two flows will be forward to node D by the aggregation rule in SDN switch B, leading to traffic congestion on link (B, D) (Fig. 1(a)). To achieve load balance, a flow rule with a longer prefix and a higher priority can be added in node B to forward flow f_{AF} to node E (Fig. 2(b)).

B. The Routing Optimization Problem With TCAM Capacity Constraint

Given the hybrid SDN-based IoT $G(V, E)$, the set F of flows, and the set P_i of feasible paths for each flow $f_i \in F$, the RO problem under the TCAM capacity constraint can be

F	The set of flows.
f_i	The ith flow in F.
P_i	The set of feasible paths of f_i.
h_i	The volume of flow f_i.
x_{ip}	A binary decision variable denotes whether the flow f_i selects feasible path $p \in P_i$.
λ	A decision variable denotes the maximum link utilization of a network.
$\gamma(v)$	The number of available TCAM entries in SDN node v.
$c(e)$	The capacity of link e.
θ_{iu}	A binary decision variable denotes whether a flow rule in SDN node u is used to adjust the forwarding path of flow f_i.
ω_{ip}^u	A binary decision variable indicates whether the feasible path p of flow f_i consumes an extra TCAM entry of SDN node u.
ξ_{ip}^e	A binary decision variable indicates whether the link e appears in the feasible path p of flow f_i.

TABLE I: The notations used in the formulation

formulated as a MILP problem. For ease of description, the notations used in the formulation are summarized in Table I.

To avoid network congestion, the optimization objective considered in this paper is Maximum Link Utilization (MLU), which is a widely used metric for load balancing [29], [32]. The RO problem tries to minimize the MLU by selecting proper forwarding paths for the flows:

$$\text{minimize} \quad \lambda \tag{1}$$

To accommodate each flow $f_i \in F$, a path must be selected from the set P_i for the flow:

$$\sum_{p \in P_i} x_{ip} = 1 \qquad \forall f_i \in F \tag{2}$$

For each SDN node $u \in V_{SDN}$, an available flow rule is required to forward flow f_i to a link $(u, v) \in E$, which is not on the shortest of flow f_i. In this case, a flow rule in SDN node u is used to adjust the forwarding path of flow f_i to a non-shortest path, and the variable θ_{iu} must be equal to 1.

$$\sum_{p \in P_i} \omega_{ip}^u x_{ip} \leq \theta_{iu} \qquad \forall f_i \in F, u \in V_{SDN} \tag{3}$$

To ensure that all the required flow rules can be installed in TCAM, the following TCAM capacity constraint for each SDN node $u \in V_{SDN}$ must be satisfied.

$$\sum_{f_i \in F} \theta_{iu} \leq \gamma(u) \qquad \forall u \in V_{SDN} \tag{4}$$

At last, we have the link utilization constraint for each link $e \in E$:

$$\sum_{f_i \in F} \sum_{p \in P_i} x_{ip} h_i \xi_{ip}^e \leq \lambda c(e) \qquad \forall e \in E \tag{5}$$

The complexity of a MILP problem is known to be exponential, i.e., $O(2^N)$, where N is the number of integer variables. Thus the MILP formulation presented above has an exponential complexity with N in $O(|V||F|)$, which makes it computationally expensive in large networks. Theorem 1 shows the problem is NP-hard.

Theorem 1: The RO problem with TCAM capacity constraint in hybrid SDN networks is NP-hard.

Proof: We now prove that the RO problem with the TCAM capacity constraint is NP-hard, by conducting a reduction from the k disjoint route problem to the RO problem with the TCAM capacity constraint. An instance of the k disjoint route problem is given by a graph $G_1(V_1, E_1)$ and a set O of k distinct node pairs. It asks for calculating k mutually link-disjoint paths for the k node pairs. This problem is known to be NP-hard [39].

We can easily conduct a reduction from the k disjoint routing problem to the RO problem with TCAM capacity constraint as follows. Let an instance $\mathcal{I}_1$ of the k disjoint problem be given by $G_1(V_1, E_1)$ and $O = \{o_1, o_2, \cdots, o_k\}$. Based on the given instance $\mathcal{I}_1$, we construct an instance I_2 of the RO problem with TCAM capacity constraint in the following way. Let $G_2(V_2, E_2) = G_1(V_1, E_1)$, and set the capacity of each link of graph $G_2(V_2, E_2)$ and the TCAM capacity of each node to 1 unit and k, respectively. For each node pair $o_i =< s_i, d_i > \in O$, we add a flow f_i between nodes s_i and d_i to flow set F. The volumes of the flows are 1 unit. Clearly, $\mathcal{I}_2$ can be constructed from $\mathcal{I}_1$ in polynomial time. It also can easily verify that the k disjoint route problem has a solution if and only the constructed RO problem has a solution with maximum link utilization 1, and the two solutions share the same paths.

Hence, to efficiently solve the RO problem in large networks, we propose approximate algorithms in Section IV.

IV. The Approximate Algorithms

We have shown that the RO problem with TCAM capacity constraint is NP-hard in Section III. B. Thus, to efficiently solve the problem, we propose approximate algorithms. In order to reduce the complexity, we solve the RO problem with TCAM constraint in two stages. In the first stage, we ignore the TCAM capacity constraint and optimize the flow routing such that the maximum link utilization is minimized. Given the paths calculated for the flows in the first stage, we need to adjust the paths of some flows to meet the TCAM capacity constraint in the second stage. For ease of description, the problems solved in the first and second stages are called Minimum Congestion Routing (MCR) and Routing Adjustment (RA) problems, respectively. In this section, we will present the approximate algorithms for solving the MCR and RA problems.

A. The Minimum Congestion Routing Problem

Given the network topology $G(V, E)$ and the set F of flows, the MCR problem can be formulated as:

$$\text{minimize} \qquad \lambda \tag{6}$$

$$\sum_{p \in P_i} x_{ip} = 1 \qquad \forall f_i \in F \tag{7}$$

$$\sum_{f_i \in F} \sum_{p \in P_i} x_{ip} \xi_{ip}^e \leq \lambda c(e) \qquad \forall e \in E \tag{8}$$

$$x_{ip} \in \{0,1\} \qquad \forall f_i \in F \tag{9}$$

The MCR problem is also NP-hard [32]. Thus, we use two approximate algorithms with performance bound to get the routing solution for the MCR problem. The two approximate algorithms for the MCR problem are called Randomized Rounding Algorithm (RRA) and Online Algorithm (OA) [40], respectively. By using the two algorithms, we can get near-optimal routing solution for the RO problem without TCAM capacity constraint.

(1) The Randomized Rounding Algorithm

In the MCR problem, the decision variable x_{ip} is a binary variable, which restricts the route of flow f_i to a single path. If we relax the binary constraint $x_{ip} \in \{0,1\}$ to $0 \leq x_{ip} \leq 1$, the relaxed MCR problem is equivalent to Maximum Concurrent Flow (MCF) problem [42]. The RRA first calculates a fractional flow routing solution for the MCF problem, and then it uses the randomized rounding strategy [41] to get a single-path routing solution for the MCR problem based on the fractional flow routing solution. **Algorithm 2** shows the detailed procedure of RRA.

In the first step (line 3), the RRA uses a Fully Polynomial Time Approximation Scheme (FPTAS) [32] (**Algoirthm 3**) to calculate routing strategy for the MCF problem. An FPTAS can guarantee that for any $\epsilon > 0$, the solution found by it has objective value with $(1+\epsilon)$-factor of the optimal, and the running time is as most a polynomial function of the network size and $1/\epsilon$. Here, the complexity of **Algoirthm 2** is $O(\epsilon^{-2}m^2 log^{O(1)}m)$, when δ is set to

$$\delta = \frac{1}{(1+n\epsilon)^{\frac{1-\epsilon}{\epsilon}}}(\frac{1-\epsilon}{m})^{\frac{1}{\epsilon}}$$

The proof for the running time can be found in [42].

In the second step (lines 4-9), RRA chooses a path for each flow $f_i \in F$ from its candidate path set calculated in the first step. Intuitively, a path carries higher traffic volume for flow f_i should be more likely to be chosen as the path for flow f_i. Therefore, RRA chooses path p_i^k as the path for flow f_i with probability $\frac{f_i^k}{\sum_{j \in P_i} f_i^j}$, where P_i is the set of candidate paths calculated for flow f_i in the first step, and f_i^j is the volume of flow f_i carried on path $p_j \in P_i$.

(2) The Online Algorithm

The FPTAS used in the RRA takes $O(\epsilon^{-2}m)$ time to calculate fractional routing solution for the MCF problem. So RRA will take a long time to get the solution if we use a small ϵ (see Section V. B). To speed up the routing calculation process, we need to use a faster algorithm, which routes each flow with only one iteration. Furthermore, the set of flows must be given in advance in RRA. However, the flows may arrive on the fly, and thus it is desired to use an Online Algorithm (OA) [40] to route every newly arrived flow.

The detailed procedure of OA is shown in **Algorithm 4**. Here, ρ is the step size of the link weight update. In the ith iteration, the OA first finds the shortest path p_i from the feasible path set P_i for flow f_i under the link weights $w = \{w(1), w(2), \cdots, w(|E|)\}$ (line 3), and then it updates the utilizations and weights of the links that are traversed by path p_i (line 4).

Algorithm 2 Randomized Rounding Algorithm

Input: Network topology $G(V,E)$, the set F of flows, and the sets $\{P_1, P_2, \cdots, P_{|F|}\}$ of feasible paths for the flows.
Output: The set P of paths for the flows, and the maximum link utilization λ.
1: $\forall e \in E$, $l_e \leftarrow 0$
2: $P \leftarrow \phi$
3: $\{FP_1, FP_2, \cdots, FP_{|F|}\} \leftarrow$ **Algorithm 3**(P, F) //Calculate the fractional routing solution for the MCF problem
4: **for** each flow $f_i \in F$ **do**
5: Choose $p_i^k \in P_i$ with probability $\frac{|f_i^k|}{\sum_{j=1}^{|P_i|}|f_i^k|}$ as the ultimate path for flow f_i
6: **if** p_i^k is chosen **then**
7: Add path p_i^k to set P
8: $\forall e \in p^i$, $l_e \leftarrow l_e + \frac{|f_i^k|}{c(e)}$
9: **end if**
10: **end for**
11: **for** each path $p_i \in P$ **do**
12: $l_{max}^i \leftarrow max_{e \in p_i} l_e$
13: **end for**
14: $\lambda \leftarrow max_{i=1}^{|P|} l_{max}^i$
15: **return** P and λ

Algorithm 3 Maximum Concurrent Flow Algorithm

Input: The set F of flows, and the sets $\{P_1, P_2, \cdots, P_{|F|}\}$ of feasible paths for the flows.
Output: The routing strategy $\{FP_1, FP_2, \cdots, FP_{|F|}\}$ for the flows.
1: $w(e) \leftarrow \frac{\beta}{c(e)}$, $\forall e \in E$
2: $f_i^j \leftarrow 0$, $\forall f_i \in F, f_i^j \in FP_i$
3: **while** $\sum_{e \in E} c(e) \cdot w(e) < 1$ **do**
4: **for** each flow $f_i \in F$ **do**
5: $h_i' \leftarrow h_i$
6: **while** $\sum_{e \in E} c(e) \cdot w(e) < 1$ and $h_i' > 0$ **do**
7: $p_i^j \leftarrow$ the shortest path in P_i with link weights w
8: $c \leftarrow min\{h_i', min_{e \in p_i^j} c(e)\}$
9: $h_i' \leftarrow h_i' - c$
10: update $f_i^j \in FP_i$ to $f_i^j + c$
11: $\forall e \in p_i^j$, $w(e) \leftarrow w(e)(1 + \epsilon \frac{c}{c(e)})$
12: **end while**
13: **end for**
14: **end while**
15: **return** $\{FP_1, FP_2, \cdots, FP_{|F|}\}$

B. The Routing Adjustment Problem

In the first phase, we solve the MCR problem without considering the TCAM capacity constraint. Thus, the paths calculated for the flows in the first phase will likely not be able to be realized in SDN networks due to the TCAM capacity constraint. In the second phase, we will adjust the paths for some flows to meet the TCAM capacity constraint. The key idea of routing adjustment is to adjust some paths requiring TCAM entries to the shortest paths. Three routing adjustment strategies, which are called Global-Optimal Strategy (GOS), Node-Optimal Strategy (NOA), and Node-Greedy Strategy (NGS), respectively, are used in our paper.

Algorithm 4 Online Algorithm

Input: Network topology $G(V, E)$, the set F of flows, and the sets $\{P_1, P_2, \cdots, P_{|F|}\}$ of feasible paths for the flows.
Output: The sets P of paths for the flows, and the maximum link utilization λ.

$l_e \leftarrow 0$, $d_e \leftarrow \frac{\beta}{c(e)}$, $\forall e \in E$
for each flow $f_i \in F$ **do**
  Choose p_i^j with minimum $\sum_{e \in p_i^j} w(e)$ as the path for flow f_i, and add path p_i^j to set P.
  $l_e \leftarrow l_e + \frac{h_i}{c(e)}$, $w(e) \leftarrow w(e)(1 + \frac{h_i}{c(e)})$, $\forall e \in p_i^j$
end for
for each $p_i \in P$ **do**
  $l_{max}^i \leftarrow max_{e \in p_i} l_e$
end for
$\lambda \leftarrow max_{i=1}^k l_{max}^i$
return P and λ

Algorithm 5 Node-Optimal Strategy

Input: The network $G(V, E)$, the set F of flows, and the set P of paths for the flows calculated in the first phase.
Output: The set P_{out} of paths for the flows.

while existing SDN switches whose TCAM capacity constraints are violated **do**
  select the SDN switch u whose TCAM capacity constraint is most seriously violated
  $F_u \leftarrow$ the set of flows that go through SDN switch u and forwarded by dedicated flow rules
  $P_u \leftarrow \{p_i | p_i \in P, f_i \in F_u\}$
  solve problem **P2**, and add the selected paths for the flows in F_u to P_{out}
end while
$P_{out} \leftarrow P_{out} \cup (P \backslash P_{out})$
return P_{out}

(1) Global-Optimal Strategy

Let p_i denote the path calculated for flow f_i in the first phase, and sp_i is the shortest path between the source and destination nodes of flow f_i. If $p_i \neq sp_i$, some extra TCAM entries will be required at some SDN switches on the path p_i to forward flow along the path p_i. To reduce TCAM usage, we can simply adjust the path of flow f_i to sp_i. However, the path adjustment may lead to load balancing performance degradation. Thus, to avoid load balancing performance degrade significantly, we need to carefully select the flows for taking path adjustment.

For ease of description, we first introduce some notations. We use variable x_i to denote whether to adjust the path p_i of flow f_i to the shortest path sp_i. The parameter ω_i^j denote whether an TCAM entry is required at node j to forward flow f_i along path p_i. Given the set F of flows and the set P_i of candidate paths for the flows, the routing adjustment problem can be formulated as the following Integer Linear Programming (ILP) problem:

$$\textbf{P1}: \quad \text{minimize} \quad \lambda \tag{10}$$

$$\sum_{f_i \in F} h_i(x_i \xi_{ip_i}^e + (1 - x_i)\xi_{isp_i}^e) \leq \lambda c(e) \qquad \forall e \in E \tag{11}$$

$$\sum_{f_i \in F} \omega_i^j x_i \leq \gamma(j) \qquad \forall j \in V_{SDN} \tag{12}$$

$$x_i \in \{0, 1\} \qquad \forall f_i \in F \tag{13}$$

Eq. (11) represents the link capacity constraint, and Eq. (12) denotes the TCAM capacity constraint. By solving the above ILP problem, we can get an optimal routing adjustment strategy for the flows. However, for large networks, the ILP problem may be intractable computationally.

(2) Node-Optimal Strategy

As presented above, the GOS optimizes the routing adjustment solution for all the flows at a time, which may lead to high computational complexity. In order to reduce the computational complexity, NOS only pursue the optimal routing adjustment solution for a part of flows at a time, and it gets the whole solution by repeating the process. Specifically, NOS first finds the SDN switch whose TCAM capacity constraint is most seriously violated, and then it optimizes the routing adjustment solution for the flows that go through the SDN switch and are forwarded by the dedicated TCAM entries at the SDN switch. For ease of description, we use F_u to denote the set of flows that go through the SDN switch u and are forwarded by dedicated TCAM entries at the SDN switch u. Similar to the GOS, the NOS also gets the optimal routing adjustment solution for the flows in F_u by solving an ILP problem. Given the set F_u of flows and the set P_u of candidate paths for the flows in F_u, the ILP is formulated as follows:

$$\textbf{P2}: \quad \text{minimize} \quad \lambda \tag{14}$$

$$\sum_{f_i \in F_u} \omega_i^u x_i \leq \gamma(u) \tag{15}$$

$$\sum_{f_i \in F_u} h_i(x_i \xi_{ip_i}^e + (1 - x_i)\xi_{isp_i}^e) + rb_e \leq \lambda c(e) \qquad \forall e \in E \tag{16}$$

$$x_i \in \{0, 1\} \qquad \forall f_i \in F_u \tag{17}$$

where rb_e is the bandwidth used by the flows in $F \backslash F_u$ on link e. Since $|F_u|$ ($u \in V_{SDN}$) is usually much smaller than $|F|$, the complexity of the ILP used by NOS is much lower than that of the ILP used by GOS. **Algorithm** 5 shows the detailed procedure of NOS.

(3) Node-Greedy Strategy

Even though the complexity of NOS is much lower than that of GOS, solving the ILP problem **P2** may also take a long time for large networks. Thus, we propose NGS, which is a greedy algorithm with polynomial time complexity.

Similar to the NOS, NGS also calculates the routing adjustment solution for the flows in a set F_u in each time. However, instead of solving an ILP, NGS uses a greedy algorithm to

Algorithm 6 Node-Greedy Strategy

Input: The network $G(V, E)$, the set F of flows, and the set P of paths for the flows calculated in the first phase.
Output: The set P_{out} of paths for the flows.

1: $P_{out} \leftarrow P$
2: **while** existing SDN switches whose TCAM capacity constraints are violated **do**
3: select the SDN switch u whose TCAM capacity constraint is most seriously violated
4: $F_u \leftarrow$ the set of flows that go through SDN switch u and forwarded by dedicated flow rules
5: $\Delta_{max} \leftarrow 1$
6: **while** the TCAM capacity of node u is not satisfied **do**
7: **for** each flow $f_i \in F_u$ **do**
8: $\Delta_i \leftarrow$ the increment of the maximum link utilization if the path of flow f_i is adjusted from p_i to sp_i
9: **if** $\Delta_i < \Delta_{max}$ **then**
10: $\Delta_{max} \leftarrow \Delta_i$
11: $f_{select} \leftarrow f_i$
12: **end if**
13: **end for**
14: $P_{out} \leftarrow P_{out} \backslash p_i$
15: $P_{out} \leftarrow P_{out} \cup sp_i$
16: **end while**
17: **end while**
18: **return** P_{out}

calculate the routing adjustment solution for the flows in a set F_u. **Algorithm** 6 shows the detailed procedure of NGS. In each iteration, NGS selects some flows going through the same SDN switch to reroute (lines 2-17). To reduce the impact of path adjustment on the maximum link utilization, NGS greedily chooses a flow to implement path adjustment at a time such that the increment of maximum link utilization is minimized (lines 7-15). In **Algorithm** 6, Δ_i represents the increment of maximum link utilization if the path of flow f_i is adjusted from p_i to sp_i (line 8).

In summary, to efficiently solve the RO problem under the TCAM capacity constraint, we use a two-stage approach to solve the problem. In the first phase, we consider the MCR problem and use RRA and OA algorithms to solve the MCR problem. In the second phase, we use the GOS, NOS, and NGS to solve the RA problem to meet the TCAM capacity constraint. Thus, we actually have six algorithms for the RO problem under the TCAM capacity constraint. For convenience, we call the six algorithms as RRA-GOS, RRA-NOS, RRA-NGS, OA-GOS, OA-NOS, and OA NGS, respectively.

V. PERFORMANCE EVALUATION

To evaluate the performance of the proposed approximate algorithms, we conduct simulations on both real ISP topologies and synthetic topologies using synthetic flows. In this section, we first describe the simulation setup, and then we show the simulation results.

A. simulation setup

Network topologies: In our simulations, we use the real ISP topologies from the Topology Zoo project [43]. There are 260 topologies in Topology Zoo [43], and we select 6 topologies of different sizes from the topology set. Table II summarizes the number of nodes and links in each topology. The selected topologies cover the small-size, medium-size, and large-size topologies. Moreover, to evaluate the performance of the algorithms on dense topologies, we also generate random topologies using Barabási-Albert (BA) and Erdös-Rényi (ER) models.

Since the real ISP and synthetic topologies do not have the link capacity information, we set the capacity of a link (u, v) base on the degrees of nodes u and v. In real networks, a link (u, v) may have a higher capacity if node u or node v has higher degree. Therefore, we set the capacity of each link $(u, v) \in E$ as follows:

1) If the degrees of both nodes u and v are higher or equal to 3, the capacity of link (u, v) is set to 39813.12 Mbps (OC 768).

2) If the degree of either node u or node v is higher or equal to 3, the capacity of link (u, v) is set to 9953.28 Mbps (OC 192).

3) Otherwise, the capacity of link (u, v) is set to 2488.32 Mbps (OC 48).

We assume that only a subset of nodes are deployed with SDN switches. let p_{SDN} denote the SDN deployment ratio, which is defined as $\frac{|V_{SDN}|}{|V|}$. We assume that the node with the higher degree has higher priority to deploy SDN switch, and the number of TCAM entries is the same for all of the SDN switches. For simplicity, the total number of TCAM entries (n) in every SDN switch is the same. In the simulations, we use r to represent the flow aggregation ratio, which is defined as the ratio between the total number of TCAM entries in each SDN switch and the total number of flows, i.e., $r = \frac{n}{|F|}$.

Flows: Since the real ISP topologies and synthetic topologies also do not have flow information, we use the gravity model to generate aggregated flow af_{ij} between each node pair $<i, j>$. In the gravity model, the volume of aggregated flow af_{ij} is as follows:

$$af_{ij} = T \frac{T_{out}(i)}{\sum_{k \in V} T^{in}(k)} \frac{T_{in}(j)}{\sum_{k \in V} T^{in}(k)}, \tag{18}$$

where $T_{out}(i)$ denotes the total volume of aggregated flows sent from node i, $T_{in}(j)$ denotes the total volume of aggregated flows injected to node j, and $T = \sum_{j \in V} T_{in}(j)$. Generally speaking, $T_{out}(i)$ and $T_{in}(i)$ is proportional to the total capacity of the links incident to node i. Thus, we have:

$$T_{in}(i) = \alpha \sum_{l \in N_i} c(l), \tag{19}$$

$$T_{out}(i) = \beta \sum_{l \in N_i} c(l), \tag{20}$$

Topology Type	Topology Name	Node Number	Link Number
Small-size	Arnes	34	47
	Cernet	41	59
Medium-size	Garr	54	71
	Dfn	58	87
Large-size	RedBestel	84	101
	VtWavenet2011	92	96

TABLE II: Real ISP topologies

where α and β are parameters randomly chosen in range [0.3, 0.8], N_i is the set of links incident to node i. In our simulations, we randomly generate aggregated flows using different α and β. Given the set of aggregated flows, we can get the near-optimal routing solution without TCAM and unsplittable flow constraints by using the maximum concurrent flow algorithm [42]. We assume that the maximum link utilization under the near-optimal routing solution is λ^*. Therefore, we can normalize the aggregated flow sizes by multiplying a factor $\frac{\theta}{\lambda^*}$ if we want to let the optimal maximum link utilization equal to θ.

The flows generated above are aggregated flows between network nodes. However, in our simulations, we need fine-grained flows, which are flows between IP prefixes. We assume that each network node has a set of IP prefixes (sub-network addresses), and the number of IP prefixes assigned to a node is uniformly distributed in [4, 5]. Let f_{ij}^k denote the kth fine-grained flow between nodes i and j. To determine the volume of flow f_{ij}^k, we use the following equation:

$$|f_{ij}^k| = |af_{ij}| \frac{len(f_{ij}^k.src)}{\sum_{pre \in Pre_i} len(pre)} \frac{len(f_{ij}^k.dst)}{\sum_{pre \in Pre_j} len(pre)} \tag{21}$$

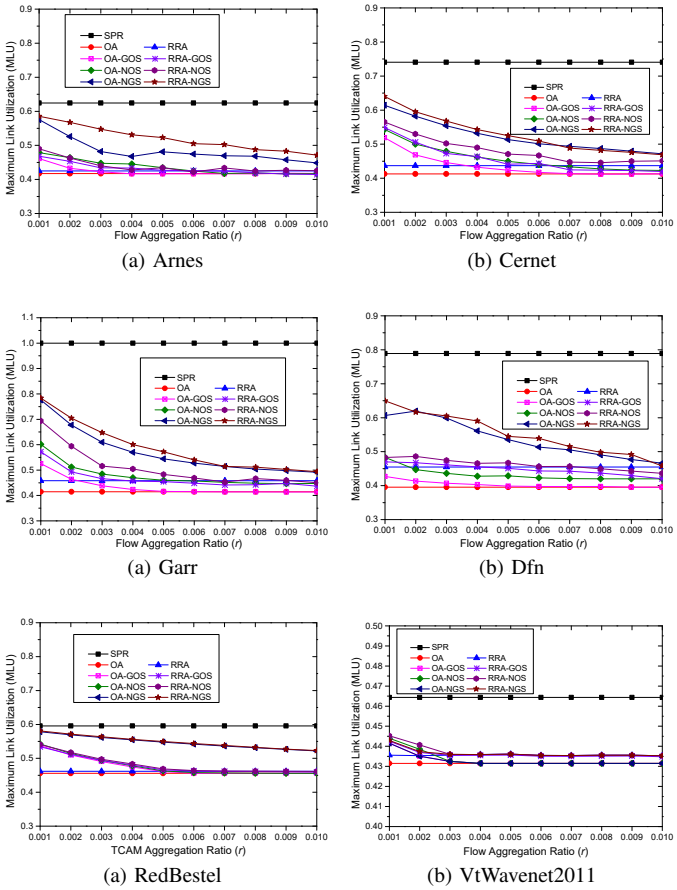
where operator $len(\cdot)$ returns the length of a prefix, and Pre_i is the set of prefixes assigned to node i.

B. simulation results

(a) Real ISP Topologies

We first present the simulation results for the six real ISP topologies chosen from Topology Zoo project [43]. Fig. 3 shows the MLUs of the algorithms under different flow aggregation ratios. In these simulations, the SDN deployment ratio is set to 1. In Fig. 3, we compare the six algorithms proposed with SPR, RRA, and OA, where SPR denotes the shortest path routing strategy. Since the flows routing along the shortest paths can be aggregated based on the destination IP prefixes, the TCAM capacity constraint is naturally met in SPR. As shown in Section IV. A, The RRA and OA only consider how to route flows to achieve the minimum MLU, and they do not consider the TCAM capacity constraint at SDN switches. Thus, the paths calculated by RRA and OA may be unrealizable in practical SDN networks. However, the results of RRA and OA can be viewed as the lower bounds for our proposed algorithms.

From Fig. 3, we have the following observations. 1) we can see that the MLUs of SPR, RRA, and OA do not vary with the flow aggregation ratio. This is because the three algorithms do not consider the TCAM capacity constraint (i.e., it is equivalent to that the flow aggregation ratio is set to 1 in the three algorithms). 2) We also can note that generally, the MLUs of our proposed algorithms decrease with the increasing of flow aggregation ratio. The reason is that as the flow aggregation ratio increases, more available TCAM entries can be used to adjust the forwarding paths of the flows, resulting in better load balance performance. 3) The load balance performance of our proposed algorithms is much better than that of SPR. This verifies that upgrading the IP networks running



(a) Arnes (b) Cernet

(a) Garr (b) Dfn

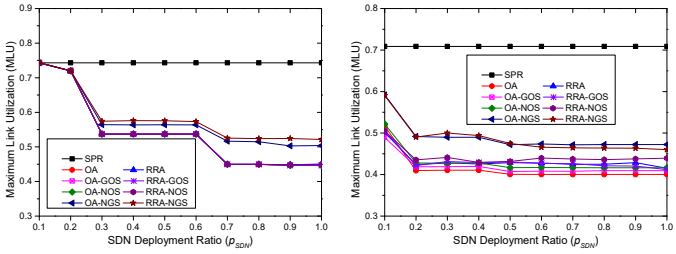
(a) RedBestel (b) VtWavenet2011

Fig. 3: The MLUs of the algorithms in real ISP topologies when flow aggregation ratio r varies

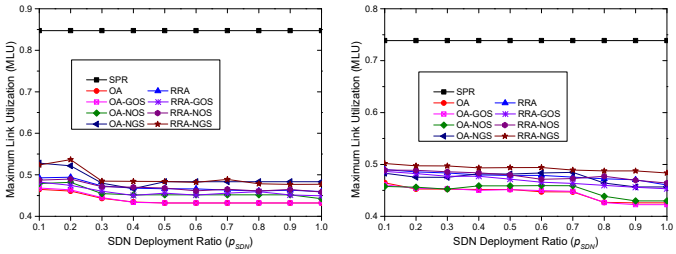
the shortest path based routing protocol to the SDN-enabled ones can significantly improve the network performance. 4) OA performs slightly better than RRA when they calculate minimum congestion routing for the flows, and as expected, the GOS and NOS have better performance than NGS when they adjust flow routing to meet the TCAM capacity constraint. 5) When the flow aggregation is higher than a threshold (e.g., 0.008 in our simulations), the performance of NGS is very close to that of GOS and NOS. 6)The performance differences of the proposed algorithms on Arnes and Cernet are obviously larger than those on RedBestel and VtWavenet. This is because Arnes and Cernet are denser (i.e., $\frac{|E|}{|V|}$ is higher) than RedBestel and VtWavenet, and thus, Arnes and Cernet have more routing optimization options than RedBestel and VtWavenet, which is beneficial for GOS and NOS. 7) Our proposed algorithms can achieve good load balancing performance even if the flow aggregation is low (e.g., $r = 0.01$), which means that we can achieve good load balancing performance with small TCAM space.

To evaluate the impact of SDN deployment ratio on the load balancing performance, we show the MLUs of the algorithms under different SDN deployment ratios in Fig. 4. In these simulations, we fix the flow aggregation ratio to 0.001. Since only the SDN switches can flexibly control the forwarding paths of the flows going through them, the MLUs of the algorithms decrease with the increasing of SDN deployment ratio. Similar to the results in Fig. 3, the load balancing

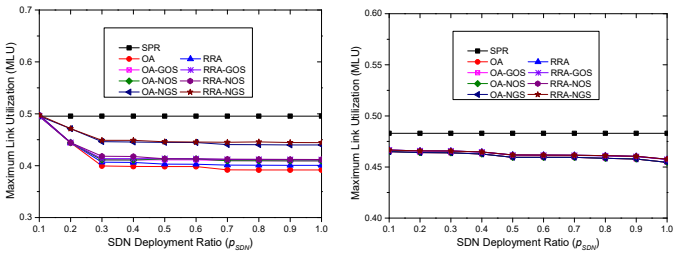
performance can be significantly improved by leveraging the flexible routing control capacity of SDN switches. From Fig. 4, we also can observe that in all cases, GOS and NOS perform better than NGS, and the performance of NOS is very close to that of GOS. Furthermore, it is notable that in most topologies, the MLUs of the algorithms decrease marginally when the SDN deployment ration is higher than 0.3. This implies that upgrading a small portion of network nodes to SDN-enable ones can achieve satisfactory performance. In addition, Fig. 4 shows that the MLUs of the GOS and NOS are very close to those of OA and RRA, which do not consider the TCAM capacity constraint. This demonstrates that by carefully optimizing the routing for the flows, we can obtain good load balancing performance even if there are only very limited number of TCAM entries in the SDN switches.



(a) Arnes (b) Cernet



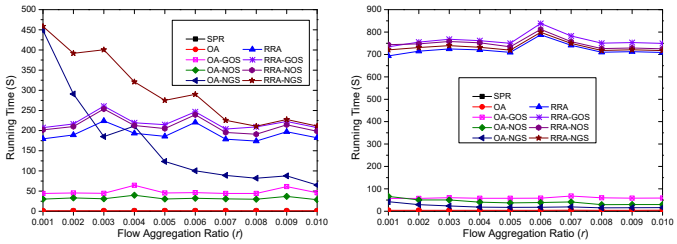
(a) Garr (b) Dfn



(a) RedBestel (b) VtWavenet2011

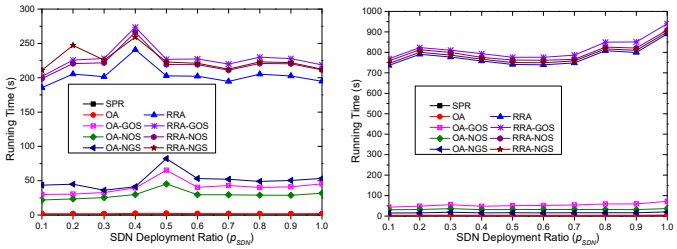
Fig. 4: The MLUs of the algorithms in real ISP topologies when the SDN deployment ratio p_{SDN} varies

Fig. 5 and Fig. 6 show the running times of the algorithms when r and p_{SDN} vary, respectively. We also have the simulation results for the other four real ISP topologies, and the results show similar trends for all the algorithms. Thus, to save space, we do not show them here. The running times of the algorithms are mainly related to the network size and the characteristics of network topologies. Generally speaking, the algorithms have longer running times on larger topologies and have shorter running times on sparser network topologies. Evidently, the running time of RRA is much longer than that of OA. The reason is that RRA uses the **Algorithm 3** to calculate the fractional flow routing solution in the first step, and the **Algorithm 3** may need a large number of iterations to get a near-optimal routing solution. From Fig. 5 and Fig. 6, we also can note that the running times of the algorithms using GOS are just slightly longer than those of the algorithms using NOS, and the running times of the algorithms using GOS may even shorter than those of the algorithms using NGS (e.g., Fig. 5(a) and Fig. 6(b)). This is because the real ISP topologies are sparse, and thus there may be only a small number of decision variables required to be determined in GOS. The results indicate that we can use GOS or NOS to get a better solution within a short time in networks with sparse topologies.



(a) Garr (b) VtWavenet2011

Fig. 5: The running times of the algorithms in real ISP topologies when flow aggregation ratio r varies



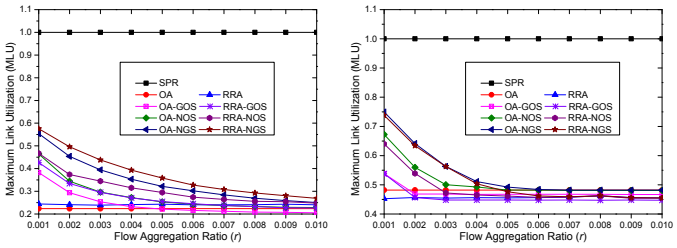
(a) Garr (b) VtWavenet2011

Fig. 6: The running times of the algorithms in real ISP topologies when the SDN deployment ratio p_{SDN} varies

(b) Synthetic Topologies

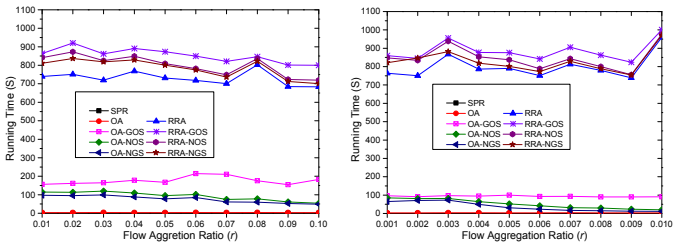
From Table. II, we know that the real ISP are sparse. To evaluate the performance of the algorithms on topologies that are denser than the real ISP topologies, we generate synthetic topologies with 70 nodes using BA and ER models and conduct simulations on these synthetic topologies. BA model generates random topologies by beginning with an initially connected topology of n_0 ($n_0 = 4$ in our simulations) nodes and adding new nodes sequentially. Each new node is connected to d_{min} existing nodes with a probability that is proportional to the degree of the existing nodes. In the ER model, a topology is constructed by independently connecting each pair of nodes by a link with a fixed probability p. In our simulations, d_{min} and p are set to 3 and 0.088, respectively, and thus the number of the links in the generated synthetic topologies is about 200.

We plot the MLUs of the algorithms in BA and ER topologies under different flow aggregation ratios in Fig. 7. From Fig. 7, we can observe that the simulation results of the algorithms in synthetic topologies and real ISP topologies



(a) BA (b) ER

Fig. 7: The MLUs of the algorithms in Synthetic topologies when flow aggregation ratio r varies



(a) BA (b) ER

Fig. 8: The running times of the algorithms in Synthetic topologies when flow aggregation ratio r varies

(Fig. 3) share similar trends. It is worth to note that the MLUs of the algorithms decrease quickly with the increasing of flow aggregation ratio, and we can get promising load balancing performance even using a small number of TCAM entries (e.g., the MLUs improve marginally when the flow aggregation ratio higher than 0.005.). Moreover, the GOS and NOS also perform better than NGS, especially when the flow aggregation is low (e.g., lower than 0.005).

We also show the running times of the algorithms in synthetic topologies in Fig. 8. Similar to the results in real ISP topologies (i.e., Fig. 5), the running times of the algorithms using RRA are much longer than those of the algorithms using OA. Furthermore, as expected, the running times of the algorithms using NGS is shorter than those of the algorithms using GOS and NOS. The reason is that as the topologies become denser, there will be more decision variables involved in the ILP models used in GOS and NOS, and thus the complexity of solving the MILP models also increase dramatically.

In summary, we can mainly get the following insights from the simulation results. 1) With the help of SDN technique, the RO performance of IP networks can be significantly improved; 2) The proposed algorithms can achieve satisfactory RO performance even with a small number of available TCAM entries in hybrid SDN networks, which may only have a small portion of SDN switches; 3) Generally, the algorithms using OA perform better than those using RRA in terms of load balancing performance and running time.

VI. Conclusion

With the rapid development of IoT and distributed cloud computing architecture, vast amounts of data collected from variety of IoT devices are required to be sent to clouds located at network nodes through backhaul or backbone network. Thus, the number of flows and the total volume of flows transmitted on the networks will increase explosively in the the near future. To guarantee the QoS of IoT cloud services and avoid network congestion, it is essential to implement efficient RO strategies for the IoT flows. On the other hand, the emergence of SDN paves a way for implementing efficient RO strategies. In SDN networks, the flow routing is realized by using flow rules usually installed in TCAM in SDN switches. However, due to physical limitations and cost constraint, the capacity of TCAM in SDN switches is very limited, and thus the number of available flows in SDN switches is far less than the number of IoT flows. Furthermore, the hybrid SDN networks, where SDN switches co-exist with traditional devices, are an important deployment scenario that needs to be considered. Therefore, in this paper, we studied the RO problem under the TCAM capacity constraint in hybrid SDN-based IoT. Specifically, we first formally formulated the problem and proved that the problem is NP-hard, and then we proposed several two-stage approximate algorithms to efficiently solve the problem. To evaluate the performance of the algorithms under the TCAM capacity constraint, we conducted extensive simulations on real ISP and synthetic topologies. The simulation results revealed that the proposed algorithms can significantly improve load balancing performance by properly leveraging the very limited number of TCAM entries in SDN switches.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 23472376, 2015.

[2] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, Convergence of MANET and WSN in IoT urban scenarios, *IEEE Sensors Journal*, vol. 13, no. 10, pp. 35583567, 2013.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case for VM-based cloudlets in mobile computing, *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 1423, 2009.

[4] S. Wang, G. Tu, R. Ganti, et al, Mobile micro-cloud: Application classification, mapping, and deployment, *in Annual Fall Meeting of ITA (AMITA)*, 2013.

[5] A. Sivanathan, D. Sherratt, H. H. Gharakheili, et al., Characterizing and classifying IoT traffic in smart cities and campuses, *IEEE INFOCOM Workshops*, 2017.

[6] A. Mendiola, J. Astorga, E. Jacob, M. Higuero, A survey on the contributions of software-defined networking to traffic engineering, *IEEE Communications Survey & Tutorials*, vol. 19, no. 2, pp. 918-953.

[7] U. D. Black, MPLS and label switching networks, *Prentice Hall*, 2002.

[8] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, The segment routing architecture, *IEEE GLOBECOM*, 2015.

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

[10] OpenFlow Switch Specification, https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[11] C. Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven wan, *ACM SIGCOMM*, pp. 15-26, 2013.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu et al. B4: Experience with a globallydeployed software defined wan, *ACM SIGCOMM*, pp. 3-14, 2013.

[13] B. Agrawal and T. Sherwood, Modeling TCAM power for next generation network devices, *IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 120129, 2006.

[14] K. Kannan and S. Banerjee, Compact TCAM: flow entry compaction in TCAM for power aware SDN, *Distributed Computing and Networking*, pp.439-444, 2013.

[15] W. Braun and M. Menth, Wildcard compression of inter-domain routing tables for openflow-based software-defined networking, *European Workshop on Software Defined Networks, 2014.*

[16] Y. Li, R. Mao, C. Kim, and M. Yu. FlowRadar: A better netFlow for data centers, *in proceedings of NSDI*, 2016.

[17] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, Z-TCAM: an SRAM-based architecture for TCAM, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 2, pp.402-406, 2015.

[18] J. Zhang, K. Xi, and H. J. Chao, Load balancing for multiple traffic matrices using SDN hybrid routing, *IEEE HPSR*, 2014.

[19] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, Dynamic scheduling of network updates, ACM SIGCOMM, 2014.

[20] Sandhya, Y. Sinha, and K. Haribabu, A survey: hybrid SDN, *Journal of Network and Computer Applications*, vol.100, 2017.

[21] R. Masoudi and A. Ghaffari, Software defined networks: a survey, *Journal of Network and Computer Applications*, vol.67, pp. 1-25, 2016

[22] B. Fortz, J. Rexford, and M. Thorup, Traffic engineering with traditional IP routing protocols, *IEEE Communication Magazine*, vol. 40, no. 10, pp.118-124, 2002.

[23] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, Traffic engineering with MPLS in the Internet, *IEEE Network*, vol. 14, no. 2, pp.28-33, 2000.

[24] M. N. Soorki and H. Rostami, Label switched protocol routing with guaranteed bandwidth and end to end path delay in MPLS networks, *Journal of Network and Computer Applications*, vol. 42, pp. 21-38, 2014.

[25] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, Optimizing network traffic engineering using segment routing, *IEEE INFOCOM*, 2015.

[26] N. Wang, K. Ho, G. Pavlou, and M. Howarth, An overview of routing optimization for Internet traffic engineeringf, *IEEE Communications Survey & Tutorials*, vo. 10, no. 1, pp.36-56, 2008.

[27] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, A survey on the contributions of software-defined networking to traffic engineering, *IEEE Communications Survey & Tutorials*, vol. 19, no. 2, pp. 918-953, 2017.

[28] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, Traffic engineering in software-defined networking: measurement and management, *IEEE Access*, vol. 4, pp. 3246-3256, 2016.

[29] M. Pioro and D. mehdi, Routing, flow, and capacity design in communication and computer networks, *Morgan Kaufmann Publishers*, 2004.

[30] S. Vissicchio, L. Vanberer, O. Bonaventure, Opportunities and research challenges of hybrid software defined networks, *ACM Computer Communication Review*, vol. 44, no. 2, 2014.

[31] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, Safe update of hybrid SDN networks, *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1649-1662, 2017.

[32] S. Agarwa, M. Kodialam, and T. V. Lakshman. Traffic engineering in software defined networks, *in proceedings of IEEE INFOCOM*, 2013.

[33] J. He and W. Song. Achieving near-optimal traffic engineering in hybrid software defined networks, *in proceedings of IFIP Networking*, 2015.

[34] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, Traffic engineering in SDN/OSPF hybrid network, *in proceedings of IEEE ICNP*, 2014.

[35] W. Wang, W. He, and J. Su. Enhancing the effectiveness of traffic engineering in hybrid SDN, *in proceedings of IEEE ICC*, 2017.

[36] S. Zhang, Q. Zhang, et al., Sector: TCAM space aware routing on SDN, *International Teletraffic Congress*, 2016.

[37] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, and H. Zhang, Incremental deployment for traffic engineering in hybrid SDN network, *International Performance Computing and Communications Conference*, 2015.

[38] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, One step at a time: optimizing SDN upgrades in ISP networks, *IEEE INFOCOM*, 2017.

[39] M. Garey, D. Johnson, Computers and Intractability, W.H. Freeman, New York, 1979.

[40] Y. Cui, B. Li, and K. Nahrstedt, On achieving optimized capacity utilization in application overlay networks with multiple competing sessions, *ACM SPAA*, 2004.

[41] P. M. Bialon, A randomized rounding approach to a k-splittable multicommodity flow problem with lower path flow bounds affording solution quality guarantees, *Telecommunication Systems*, pp. 525-542, 2017.

[42] G. Karakostas, Faster approximation schemes for fractional multicommodity flow problems, *ACM/SIAM SODA*, 2002.

[43] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, The Internet Topology Zoo, *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 9, pp. 1765 - 1775, 2011.