Joint Routing and Sketch Configuration in Software-Defined Networking

Yutong Zhai, Hongli Xu, *Member, IEEE*, Haibo Wang, *Student Member, IEEE*, Zeyu Meng, and He Huang, *Member, IEEE*

***Abstract*— Traffic measurement is very important for various applications, such as traffic engineering and attack detection, in software-defined networks. Due to limited resources (*e.g.*, computing, memory) on SDN switches, sketches have been widely used for efficient traffic measurement. Meanwhile, multiple independent sketches are required for different application requirements, such as heavy hitter detection and flow size distribution estimation. To avoid the measurement redundancy, we configure which sketch(es) will measure flow on each switch. If traffic measurement is performed on each switch without considering network routing, due to traffic dynamics, it will cause massive measurement overhead on a switch, which may exceed the switch's computing capacity. In this paper, we study the joint optimization of flow routing and sketch configuration in SDNs. We formulate this problem as an integer linear programming and prove its NP-hardness. To solve this problem, we propose a rounding-based offline algorithm and a primal-dual-based online algorithm for different application scenarios. To deal with bursty traffic, we also design an adaptive sampling mechanism for high-fidelity traffic measurement. We formally analyze the approximation performance or competitive ratio of the proposed algorithms. The extensive simulation results show the high efficiency of our proposed algorithms. For example, the online algorithm can improve the network throughput 30% compared with the state-of-the-art.**

***Index Terms*— Software defined networks, traffic measurement, approximation, flow routing, sketch configuration.**

I. Introduction

SOFTWARE-DEFINED networking (SDN) is a new paradigm that separates the control plane and the data plane on the independent devices in a network [1], [2]. The control plane, consisting of one or a cluster of controllers, can provide logically centralized and fine-grained control for flow management and optimize network resource utilization [3]. To explore full advantages and improve the performance (*e.g.*, load balancing, throughput maximization) of SDN, the traffic statistics of 5-tuple (or fine-grained) flows play an important role in various network applications, such as network monitoring [4], traffic engineering [5], [6], load balancing [7], network function virtualization [8], security detection [9], and QoS routing [10].

Under the SDN framework, there are three common methods for traffic measurement. 1) The first solution is using the flow tables based on ternary content addressable memory (TCAM) on SDN switches [11], [12]. TCAM is a special type of high-speed memory, which can lookup the entire memory space within a single clock cycle. As specified by OpenFlow [13], each flow entry is able to count the traffic amount of the matched flows. However, due to high price and power consumption, the size of a TCAM-based flow table is usually limited. For example, even the high-end Broadcom Trident2 chipset supports only 1.6×10^4 forwarding rules [14]. Though the software switch contains more flow entries than the hardware switches, it will occur massive control overhead for installing and updating a large number of rules, especially in a large-scale virtual network. In the data center, there are always too many flows (*e.g.*, $\geq 10^6$) [15]. To serve all flows, the aggregate path should be applied [16]. Since many flows may match one flow entry, their traffic amount will be aggregated accordingly. Therefore, we are not able to derive the statistics information of each individual flow by TCAM-based flow tables. 2) The second general way for traffic measurement is packet sampling. One well-known solution is Cisco Netflow [17], which allows IP flow level measurement. Another popular traffic sampling is sFlow [18]. Similar to the OpenFlow standard, sFlow keeps statistics on all interfaces. However, due to the computing resource constraint on SDN switches, traffic sampling cannot measure all packets of each flow. Therefore, the accuracy of traffic sampling is low and only supports coarse-grained measurement [19]. 3) Different from the above two measurement methods, the third one is using sketches, which can provide fine-grained traffic statistics for individual flows. Sketches are special data structures that use Hash functions to summarize traffic statistics of all 5-tuple flows with bounded error using small memory space [20]. In a high-speed network environment, sketches can store traffic characteristics and only take up less memory resources on switches.

Recently, sketch-based solutions [20], [21] have been widely used for traffic measurement [22], [23]. For example, CountMax [24] is designed for detecting the maximum k (*i.e.*, top-k) flows in the network. OpenSketch can be used to discover

Manuscript received May 20, 2019; revised December 25, 2019 and May 2, 2020; accepted June 10, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Malone. Date of publication July 2, 2020; date of current version October 15, 2020. The work of Yutong Zhai, Hongli Xu, Haibo Wang, and Zeyu Meng was supported in part by the National Science Foundation of China (NSFC) under Grant 61822210, Grant U1709217, and Grant 61936015 and in part by the Anhui Initiative in Quantum Information Technologies under Grant AHY150300. The work of He Huang was supported by the NSFC under Grant 61873177. *(Corresponding authors: Hongli Xu; He Huang.)*

Yutong Zhai and Hongli Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with the Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China (e-mail: zyt1996@mail.ustc.edu.cn; xuhongli@ustc.edu.cn).

Haibo Wang was with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China. He is now with the Department of Computer and Information Science and Technology, University of Florida, Gainesville, FL 32611 USA (e-mail: wanghaibo@ufl.edu).

Zeyu Meng is with the School of Cyberspace, University of Science and Technology of China, Hefei 230027, China, and also with the Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China (e-mail: sa516212@mail.ustc.edu.cn).

He Huang is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (e-mail: huangh@suda.edu.cn).

Digital Object Identifier 10.1109/TNET.2020.3002783

DDoS attacks and superspread attacks [25]. For the flow size distribution estimation, MRAC [26] is proposed to infer the usage pattern of the network. However, as network management and applications become more complex and diverse, a single sketch is difficult to implement all measurement tasks. Therefore, to fulfill the diverse network measurement tasks, it is necessary to deploy multiple sketches on SDN switches [23], [25].

In this paper, we consider the sketch-based traffic measurement in a modern virtual cloud [27], [28], which consists of many software switches, such as OVS [29], Microsoft Hyper-V virtual switches [30], and Cisco Nexus 1000V virtual switches [31]. When multiple sketches are deployed on each SDN software switch, another serious problem may occur. In general, software switches have limited computing capacity due to low-speed processing capacity. For example, an OVS switch is equipped with Intel Xeon(R) E5345 2.33GHz CPU [32]. On the other hand, with the development of information technology, the network faces more and more traffic [33]. When too much traffic passes through a switch, it will face with very heavy computing overhead. As a result, other basic functions (*e.g.*, flow-table update) on this switch may be interfered. To overcome this challenge, it is required to balance the computing overhead of traffic measurement among all switches in the network.

The traffic measurement overhead on a switch mainly depends on two factors. The first factor is the amount of traffic through a switch, which is largely determined by *flow routing*. The second factor is the average computing overhead per packet for traffic measurement. On arriving at a switch, if this packet is measured by all the sketches deployed on this switch, it may lead to measurement redundancy. In other words, each packet may be measured by one type of sketches more than one time. Thus, it is another way to reduce the computing overhead by *sketch configuration*, that is, determining by which sketch(es) one flow will be processed on a switch.

In this paper, we study the joint optimization of routing and sketch configuration in SDNs, which is the first work to consider the trade-off optimization between measurement overhead and network throughput. In our work, the SDN controller assigns the routing path for each individual flow and configures different sketches on switches for traffic measurement. The idea of joint optimization holds great promise of solving the dilemma between measurement efficiency and network performance. As another challenge, to further reduce the measurement overhead with traffic burstiness, we design an adaptive sampling mechanism. The main contributions of this paper are as follows:

1) We first propose a novel framework for flow routing and sketch-based measurement in SDNs. Then we describe the joint routing and sketch configuration problem named RSC as an integer programming and prove its NP-hardness.

2) To solve RSC, we propose two efficient algorithms called R-RSC and PD-RSC for the offline and online versions, respectively. We also analyze the approximation factor and the competitive ratio for the proposed algorithms.

3) To deal with traffic burstiness, we design an adaptive packet sampling mechanism, called QTPS, based on queuing theory for high-fidelity traffic measurement to avoid the measurement congestion.

4) The simulation results demonstrate the high efficiency of our proposed algorithms. For example, PD-RSC improves network throughput by about 30% compared with the state-of-the-art. Moreover, the measurement error of QTPS is only 1/6 as that of the existing methods.

TABLE I
KEY NOTATIONS

Symbol	Semantics
V	a set of SDN switches
E	a set of links
Γ	a set of macroflows
$\mathcal{S}$	a set of sketches
$t(s_k)$	the computing overhead per packet of sketch s_k
$N(\gamma_i)$	the number of packets in macroflow γ_i
$f(\gamma_i)$	the traffic size of macroflow γ_i
P_{γ_i}	a set of feasible paths for macroflow γ_i
$y_{\gamma_i}^p$	γ_i will be routed on path p or not
$x_{i,j}^k$	traffic in γ_i will be measured by sketch s_k on switch v_j or not
$\mathcal{I}_p^{v_j}$	switch v_j lies on path p or not
$c(e)$	the load capacity of link e
$d(v_j)$	the reserved computing capacity for measurement on switch v_j
$\mathcal{H}$	a set of routing and measurement schemes

II. PRELIMINARIES

A. Network and Sketch Models

An SDN typically consists of a logically-centralized controller and a set of switches. These switches comprise the data plane of an SDN. Thus, we model the network topology of the data plane as a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of SDN switches and E represents the communication links among these switches. Note that, compared with a centralized physical controller, distributed controllers have been proposed to improve the scalability and reliability in SDNs [34]. Since we focus on the resource (*e.g.*, CPU, bandwidth) consumption in the data plane, the controller plane will not impact these metrics significantly. For ease of presentation, we thus assume only one controller in an SDN. In Table I, we list Some key notations.

Different network applications require various sketches for traffic measurement. For example, Flowradar [19] can detect the heavy hitters, while MRAC [26] estimates the flow size distribution. To support diverse application requirements, we assume that there deploys a set of sketches, denoted as $\mathcal{S} = \{s_1, \ldots, s_q\}$ with $q = |\mathcal{S}|$, on each SDN switch. In practice, many commodity sketches are equipped with SRAM of 50-100MB [35]. In fact, the memory cost for a sketch is about 1MB on average. For example, Flowradar [19], Twolevel [25], and MRAC [26] are deployed on an SDN switch. The memory cost of these sketches is 1.7MB [19], 0.9MB [25] and 0.26MB [26], respectively. The total memory cost is not more than 3MB. Thus, it is enough to accommodate multiple sketches on an SDN switch. These sketches will estimate different packet header combinations of interest and provide fine-grained measurement. For sketch s_k, we use its average computing overhead per packet, denoted as $t(s_k)$, through long-term measurement.

B. Packet Processing for Routing and Measurement

The overall design of packet processing for joint routing and measurement on an SDN switch is illustrated through

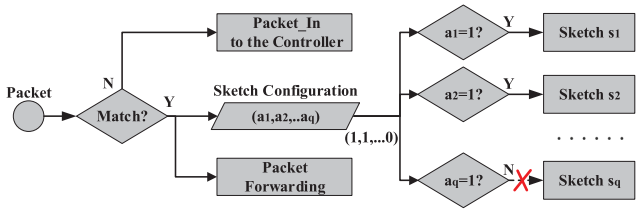


Fig. 1. Illustration of packet processing for joint flow routing and traffic measurement.

a flow chart in Fig. 1. When a packet arrives at a switch, if no matching flow entry exists, this switch will request a new rule from the controller. Otherwise, the switch will take the following operations. 1) This packet will be forwarded as specified by the matching flow entry. 2) The switch obtains the sketch configuration scheme, which consists of a tuple with q elements, denoted as $(a_1, a_2, \ldots, a_q)$, from the matching flow entry. Each element a_i in the tuple indicates whether sketch s_i will measure this flow or not. 3) The switch buffers the packet header, combined with the sketch configuration scheme, into the queue. When this packet becomes the header element of the queue, it will be measured by different sketches according to the sketch configuration scheme. For example, $a_1 = 1$ represents sketch s_1 will measure this flow and $a_q = 0$ represents sketch s_q will not.

We should note that, since the measurement rate (*i.e.*, the traffic amount measured by a switch per second) is usually slower than the forwarding rate, the queue may be full, especially facing the bursty traffic. A natural way for this situation is to directly drop packets, which will reduce the traffic measurement accuracy. To this end, we design an adaptive sampling mechanism for high-fidelity traffic measurement in Section V.

C. Joint Routing and Sketch Configuration (RSC)

To reduce the flow entry cost and alleviate the control overhead, network routing is based on not the fine-grained 5-tuple flows but coarse-grained flow granularity. Note that sketches can measure the traffic size of 5-tuple (fine-grained) flows. In this paper, we regard that traffic belonging to an origin-destination (OD) pair is aggregated as a macroflow [36]. We denote the macroflows as $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$, with $|\Gamma| = m$. Through long-term traffic observation, we can estimate the current traffic size $f(\gamma_i)$ and the number of packets $N(\gamma_i)$ for each macroflow γ_i. In fact, we install rules for the granularity of OD pairs (or macroflows), and the counter field in each flow entry can record the traffic size of this matched macroflow. The controller can obtain the traffic information of each macroflow using the lightweight traffic statistics collection solution [37]. To estimate the traffic size of each OD pair, we can adopt the commonly used solutions, such as least square methods [38], simulated annealing [39] and Kalman filters [40], for different applications.

We use $\mathcal{P}_{\gamma_i}$ to represent a set of feasible paths from source to destination of each macroflow $\gamma_i \in \Gamma$, which is pre-computed based on the network topology and dynamically updated only if the network topology is changed [41]. For each macroflow γ_i, we will choose one feasible path p from $\mathcal{P}_{\gamma_i}$ as its routing path, and install rules on switches along this routing path. Each rule also specifies the sketch(es) that will measure the traffic in this macroflow on a switch. We consider the following three constraints. 1) Since the switch needs to perform rule operations on the flow-table, we expect only a given fraction (*e.g.*, 1/3) of the switch's computing capacity is reserved for traffic measurement. Thus, to achieve load balancing, the computing overhead for traffic measurement on switch v_j should not exceed $\lambda \cdot d(v_j)$, where λ is the load balancing factor, and $d(v_j)$ is its reserved computing capacity for measurement. 2) The number of occupied flow entries on each switch v_j should not exceed a threshold $s(v_j)$, which denotes the number of permitted rules on switch v_j, also called rule-table threshold. In fact, if the controller installs a large number of rules on software switches, the control overhead for rule installment and modification will be increasing. Sometimes, it will take an unacceptable delay to install a large number of rules on a software switch [42]. 3) For each link e, its traffic load should not exceed $\lambda \cdot c(e)$, where $c(e)$ is its link capacity. Accordingly, we formalize the joint routing and sketch configuration (RSC) problem as follows:

$$\min \ \lambda$$
$$S.t. \begin{cases} \sum\limits_{p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} = 1, & \forall \gamma_i \in \Gamma \\ \sum\limits_{v_j \in p} x^k_{i,j} \geq y^p_{\gamma_i}, & \forall p, \gamma_i, s_k \\ \sum\limits_{v_j \in V} \sum\limits_{p \in \mathcal{P}_{\gamma_i}} \mathcal{I}^{v_j}_p \cdot x^k_{i,j} \geq 1, & \forall \gamma_i, s_k \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{e \in p : p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} \cdot f(\gamma_i) \leq \lambda \cdot c(e), & \forall e \in E \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{s_k \in \mathcal{S}} x^k_{i,j} \cdot N(\gamma_i) \cdot t(s_k) \leq \lambda \cdot d(v_j), & \forall v_j \in V \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{v_j \in p : p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} \leq s(v_j), & \forall v_j \in V \\ y^p_{\gamma_i} \in \{0, 1\}, & \forall p, \gamma_i \\ x^k_{i,j} \in \{0, 1\}, & \forall \gamma_i, v_j, s_k \end{cases} \tag{1}$$

In Eq. (1), $y^p_{\gamma_i}$ denotes whether macroflow γ_i will be routed on path p or not, and $x^k_{i,j}$ represents whether traffic in γ_i will be measured by sketch s_k on switch v_j or not. The first set of equations means that each macroflow will be routed through one path in the feasible path set. The second set of inequalities means that all traffic in one macroflow will be measured only on those forwarding switches. The third set of inequalities expresses the traffic measurement requirement for each macroflow, where $\mathcal{I}^{v_j}_p$ denotes whether switch v_j lies on path p or not. The fourth and fifth sets of inequalities mean the traffic load on link e and computing overhead on switch v_j. The sixth set of inequalities denotes the rule-table threshold constraint on each switch. Our goal is to achieve load balancing, that is, $\min \ \lambda$.

Theorem 1: The RSC problem is NP-hard.

Proof: We consider a simplified version of RSC, in which there is no constraint on the computing capacity and the rule-table threshold on switches. Then the simplified-RSC becomes an unsplittable multi-commodity flow problem with minimum congestion [43]. Since the multi-commodity flow problem is NP-hard, RSC is NP-hard too. □

D. A Motivating Example

Here we present a toy example to illustrate the necessity of RSC. In Fig. 2, a network includes four switches $\{v_1, v_2, v_3, v_4\}$ and two macroflows $\{F_1, F_2\}$. The gray circle and the solid black line indicate the switch and data link, respectively. Both the link bandwidth and switch computing capacity are set as 1000. For each macroflow, both the flow size and the number of packets are set as 300. We assume

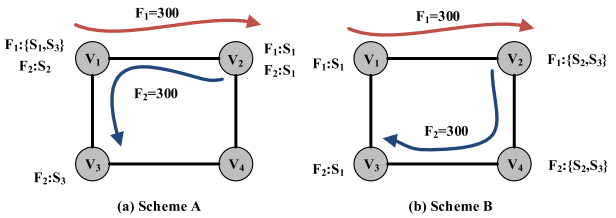


Fig. 2. A network scenario includes four switches $\{v_1, v_2, v_3, v_4\}$ and two macroflows $\{F_1, F_2\}$. The tuple of a switch indicates the sketch configuration for each flow on this switch. For example, the tuple $F_1 : \{s_2, s_3\}$ around switch v_2 in the right plot denotes that v_2 will measures macroflow F_1 by sketches s_2 and s_3. The left plot shows an initial routing and measurement scheme, and the right plot shows the optimal routing and sketch configuration scheme.

that each macroflow is measured by three types of sketches $\{s_1, s_2, s_3\}$, and the computing overhead per packet of these sketches is $1.0, 0.5, 0.5$, respectively.

The left plot of Fig. 2 shows the initial routing and measurement scheme (also called scheme A). Under scheme A, macroflow F_1 follows path $v_1 \rightarrow v_2$ and macroflow F_2 selects path $v_2 \rightarrow v_1 \rightarrow v_3$. We find that the most-congested data link is $v_1 - v_2$, and the switch with the highest measurement overhead is v_2. The traffic load of $v_1 - v_2$ equals to $300+300 = 600$, and the measurement overhead of switch v_2 equals to $300 * (0.5 + 0.5) + 300 * 1.0 = 600$. Since both the maximum traffic load and the highest measurement overhead equal to 600, the load balancing factor is 0.6. However, by jointly considering the flow routing and sketch configuration (also called scheme B), we can decrease the optimal load balancing factor to 0.3, which has an improvement of 50%, as shown in the right plot of Fig. 2. Under scheme B, macroflow F_1 also follows path $v_1 \rightarrow v_2$ and macroflow F_2 selects path $v_2 \rightarrow v_4 \rightarrow v_3$. We adjust the sketch configuration to balance the measurement overhead on all switches. We find that both the maximum traffic load and the highest measurement overhead equal to 300. As a result, RSC can achieve both link load and traffic measurement balance.

III. Offline Algorithm for RSC

A. Algorithm Description

To solve the problem in Eq. (1), we first constructs linear programming as relaxation of RSC. More specifically, RSC assumes that the controller will choose one routing path for each macroflow, and a macroflow will be measured by each type of sketch. By relaxing these assumptions, each macroflow $\gamma_i \in \Gamma$ is splittable and can be routed through a feasible path set $\mathcal{P}_{\gamma_i}$, and to be measured by several sketches with one type. We formulate the linear programming LP as follows.

$$\min \ \lambda$$

$$S.t. \begin{cases} \sum\limits_{p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} = 1, & \forall \gamma_i \in \Gamma \\ \sum\limits_{v_j \in p} x^k_{i,j} \geq y^p_{\gamma_i}, & \forall p, \gamma_i, s_k \\ \sum\limits_{v_j \in V} \sum\limits_{p \in \mathcal{P}_{\gamma_i}} \mathcal{I}^{v_j}_p \cdot x^k_{i,j} \geq 1, & \forall \gamma_i, s_k \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{e \in p : p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} \cdot f(\gamma_i) \leq \lambda \cdot c(e), & \forall e \in E \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{s_k \in \mathcal{S}} x^k_{i,j} \cdot N(\gamma_i) \cdot t(s_k) \leq \lambda \cdot d(v_j), & \forall v_j \in V \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{v_j \in p : p \in \mathcal{P}_{\gamma_i}} y^p_{\gamma_i} \leq s(v_j), & \forall v_j \in V \\ y^p_{\gamma_i} \in [0, 1] & \forall p, \gamma_i \\ x^k_{i,j} \in [0, 1] & \forall \gamma_i, v_j, s_k \end{cases} \tag{2}$$

Note that variables $y^p_{\gamma_i}$ and $x^k_{i,j}$ are integral in Eq. (1), but fractional in Eq. (2). Since LP is linear programming, we can use a linear programming solver to solve it in polynomial time. Assume that the optimal solution for LP is denoted as $\{\widetilde{y}^p_{\gamma_i}, \widetilde{x}^k_{i,j}\}$, and the optimal result is denoted as $\widetilde{\lambda}$. As LP is relaxation of RSC, $\widetilde{\lambda}$ is the lower-bound result for RSC. Using the randomized rounding method [44], we derive the integral solution, denoted by $\{\widehat{y}^p_{\gamma_i}\}$, for $\forall \gamma_i \in \Gamma$ and $\forall p \in \mathcal{P}_{\gamma_i}$. If $\widehat{y}^p_{\gamma_i} = 1$, it means that macroflow γ_i will be forwarded through path p. For macroflow γ_i, assume that the selected path for this macroflow is p. For each sketch s_k, we compute the total weight as $\delta_{i,k} = \sum_{v_j \in p} \widetilde{x}^k_{i,j}$. Based on the second set of inequalities in Eq. (2), it follows $\delta_{i,k} \geq \widetilde{y}^p_{\gamma_i}$, for $\forall \gamma_i \in \Gamma$ and $\forall s_k \in S$. Then we compute the variables $\bar{x}^k_{i,j} = \frac{\widetilde{x}^k_{i,j}}{\delta_{i,k}}$, and derive the integral solution, denoted as $\{\widehat{x}^k_{i,j}\}$, using randomized rounding. R-RSC is formally described in Alg. 1.

We should note that each macroflow will be assigned one feasible path for packet forwarding by R-RSC. In the following, we illustrate the rounding procedure. For example, there are three feasible paths for macroflow γ_i, and the optimal solution for the three feasible paths equals $\{0.2, 0.3, 0.5\}$. Then the interval $[0, 1]$ is splitted into three parts: $(0, 0.2]$, $(0.2, 0.5]$ and $(0.5, 1]$. We generate a random value between 0 to 1, and choose a path depends on this value. If the value is less than 0.2, the SDN controller will choose the first feasible path as the routing path for this macroflow. Otherwise, if the value is larger than 0.2 and less than 0.6, then the controller will set the second feasible path as the routing path. Meanwhile, if the value is larger than 0.6, the last feasible path will be set as the routing path.

Algorithm 1 R-RSC: Rounding-Based Algorithm for RSC

1: **Step 1: Solving the relaxed RSC problem**
2: Construct a linear programming LP in Eq. (2)
3: Derive the optimal solution $\widetilde{y}^p_{\gamma_i}$ and $\widetilde{x}^k_{i,j}$
4: **Step 2: Routing and Sketch Configuration**
5: Obtain an integer solution $\widehat{y}^p_{\gamma_i}$ by randomized rounding
6: **for** each macroflow $\gamma_i \in \Gamma$ **do**
7: **if** $\widehat{y}^p_{\gamma_i} == 1$ **then**
8: **for** each switch v_j along path p **do**
9: **for** each sketch s_k on switch v_j **do**
10: $\delta_{i,k} = \sum_{v_j \in p} \widetilde{x}^k_{i,j}$
11: $\bar{x}^k_{i,j} = \frac{\widetilde{x}^k_{i,j}}{\delta_{i,k}}$
12: Obtain an integer solution $\widehat{x}^k_{i,j}$ by randomized rounding
13: Install a flow entry, combining with sketch configuration $(\widehat{x}^1_{i,j}, \widehat{x}^2_{i,j} \ldots, \widehat{x}^q_{i,j})$

B. Performance Analysis

First, we give an overview that R-RSC can achieve the bi-partite approximation performance with a high probability for large-scale networks. Then we consider the approximation performance under a special case. Assume that the minimum capacity of all links is denoted as $c_{\min}$. Before performance analysis, we define a variable ϖ to assist our proof as follows:

$$\varpi = \min\{s(v_j), \forall v_j \in V; \frac{\widetilde{\lambda} \cdot c_{\min}}{f(\gamma_i)}, \forall \gamma_i \in \Gamma; \frac{\widetilde{\lambda} \cdot d(v_j)}{N(\gamma_i) t(s_k)}, \forall \gamma_i \in \Gamma, v_j \in V, \ s_k \in \mathcal{S}\} \tag{3}$$

On the right side of Eq. (3), the first item $s(v_j)$ represents the rule-table threshold of switch v_j; the second item $\frac{\widetilde{\lambda}\cdot c_{\min}}{f(\gamma_i)}$ denotes the ratio of the minimum link capacity to the traffic intensity of macroflow γ_i; and the last item $\frac{\widetilde{\lambda}\cdot d(v_j)}{N(\gamma_i)t(s_k)}$ represents the ratio of the available computing capacity on switch v_j to the computing overhead of marcoflow γ_i measured by sketch s_k. In most situations, the intensity of most macroflows is much less than the link capacity, and the measurement overhead of one macroflow is also much less than the computing capacity on SDN switches. Moreover, the rule-table threshold on a switch reaches several thousands. Therefore, it follows that $\varpi \gg 1$. We first give a general bound of R-RSC.

Theorem 2: The approximation ratio of the R-RSC algorithm is $(O(\frac{\log n}{\varpi}+1), O(\frac{\log n}{\varpi}+1))$, where n is the number of switches in the network.

Proof: Similar to the proof in [7], we can prove that R-RSC can achieve an objective value within $O(\frac{\log n}{\varpi}+1)$ compared with the optimal result $\widetilde{\lambda}$ and each resource constraint is violated at most $O(\frac{\log n}{\varpi}+1)$ multiplicatively. □

Furthermore, under some specific situations, R-RSC can achieve near-optimal performance with a high probability. Assume that the link capacity on link e, the computing capacity on switch v_j and the rule-table threshold on switch v_j are at least linear with n. That is, $\varpi = \Omega(n)$, where ϖ is defined in Eq. (3). We divide the proof into several subparts, link capacity constraint, rule-table threshold constraint, and switch computing capacity constraint.

1) Link Capacity Performance: Before analysing the link resource performance, we define a random variable $m^e_{\gamma_i}$ to denote the traffic load on link e from macroflow γ_i.

Definition 1: For each macroflow γ_i and each link e, a random variable $m^e_{\gamma_i}$ is defined as:

$$m^e_{\gamma_i} = \begin{cases} f(\gamma_i), & \text{with probability of } \sum\limits_{p:e\in p, p\in\mathcal{P}_{\gamma_i}} \widetilde{y}^p_{\gamma_i} \\ 0, & otherwise. \end{cases} \tag{4}$$

Lemma 3: After the rounding process, for any $\epsilon > 0$, the load of each link will not exceed $\widetilde{\lambda}c(e)$ by a factor of $\epsilon+1$ with a probability close to 1. In other words, for each variable n with sufficiently large value, there exists a variable $\nu > 0$, and we have

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma}\frac{m^e_{\gamma_i}}{\widetilde{\lambda}c(e)} \le (1+\epsilon)\right] \ge 1 - exp(-\nu n) \tag{5}$$

Proof: $\widetilde{\lambda}c(e)$ represents the load on link e in the optimal case. According to Definition 1, $m^e_{\gamma_1}$, $m^e_{\gamma_2}\ldots$ are mutually independent. $\sum_{\gamma_i\in\Gamma} m^e_{\gamma_i}$ denotes the actual load on link e after the rounding process. Then it follows

$$\begin{aligned}\mathbb{E}\left[\sum_{\gamma_i\in\Gamma} m^e_{\gamma_i}\right] = \sum_{\gamma_i\in\Gamma}\mathbb{E}[m^e_{\gamma_i}] &= \sum_{\gamma_i\in\Gamma}\sum_{p:e\in p, p\in\mathcal{P}_{\gamma_i}} \widetilde{y}^p_{\gamma_i}\cdot f(\gamma_i) \\ &\le \widetilde{\lambda}\cdot c(e)\end{aligned} \tag{6}$$

Combining Eq. (6) and the definition of ϖ in Eq. (3), we have

$$\begin{cases} \frac{m^e_{\gamma_i}\cdot\varpi}{\widetilde{\lambda}c(e)} \in [0,1] \\ \mathbb{E}\left[\sum_{\gamma_i\in\Gamma}\frac{m^e_{\gamma_i}\cdot\varpi}{\widetilde{\lambda}c(e)}\right] \le \varpi. \end{cases} \tag{7}$$

By applying chernoff bound, assume that ϵ is an arbitrary positive value. It follows

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma}\frac{m^e_{\gamma_i}\cdot\varpi}{\widetilde{\lambda}c(e)} \ge (1+\epsilon)\varpi\right] \le e^{\frac{-\epsilon^2\varpi}{2+\epsilon}} \tag{8}$$

Since $\varpi = \Omega(n)$, we can find some $C' > 0$, such that $\varpi = C'n$. Eq. (8) can be simplified as

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma}\frac{m^e_{\gamma_i}\cdot\varpi}{\widetilde{\lambda}c(e)} \ge (1+\epsilon)\varpi\right] \le e^{\frac{-\epsilon^2 C' n}{2+\epsilon}} \tag{9}$$

Let ν be $\frac{-\epsilon^2\ C'}{2+\epsilon}$. We have

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma}\frac{m^e_{\gamma_i}\cdot\varpi}{\widetilde{\lambda}c(e)} \ge (1+\epsilon)\varpi\right] \le exp(-\nu n) \tag{10}$$

which gives the desired inequality. □

2) Rule-Table Threshold Performance: Similarly, we define a random variable $g^{v_j}_{\gamma_i}$ to denote the number of flow entries used for macroflow γ_i on switch v_j.

Definition 2: For each macroflow γ_i and each switch v_j, a random variable $g^{v_j}_{\gamma_i}$ is defined as:

$$g^{v_j}_{\gamma_i} = \begin{cases} 1, & \text{with probability of } \sum\limits_{p:v_j\in p, p\in\mathcal{P}_{\gamma_i}} \widetilde{y}^p_{\gamma_i} \\ 0, & otherwise. \end{cases} \tag{11}$$

Lemma 4: After the rounding process, for any $\rho > 0$, the number of flow entries on each switch will not exceed $s(v_j)$ by a factor of $\rho+1$ with a probability close to 1. In other words, for each variable n with sufficiently large value, there exists a variable $\eta > 0$, and we have

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma}\frac{g^{v_j}_{\gamma_i}}{s(v_j)} \le (1+\rho)\right] \ge 1 - exp(-\eta n) \tag{12}$$

we can prove Lemma 4 in a similar way to the proof of Lemma 3. Thus, we omit the detailed proof here.

3) Switch Computing Capacity Performance: Furthermore, we define a random variable $q^{v_j}_{\gamma_i,s_k}$ to denote the amount of computing overhead for sketch s_k on switch v_j from macroflow γ_i.

Definition 3: For each macroflow γ_i, each switch v_j and each sketch s_k, a random variable $q^{v_j}_{\gamma_i,s_k}$ is defined as:

$$q^{v_j}_{\gamma_i,s_k} = \begin{cases} t(s_k)\cdot N(\gamma_i), & \text{with probability of } \bar{x}^k_{i,j} \\ 0, & otherwise. \end{cases} \tag{13}$$

Lemma 5: After the rounding process, for any $\varsigma > 0$, the computing overhead on each switch will not exceed $\widetilde{\lambda}d(v_j)$ by a factor of $\varsigma+1$ with a probability close to 1. In other words, for each variable n with sufficiently large value, there exists a variable $\chi > 0$, and we have

$$\mathbf{Pr}\left[\sum_{\gamma_i\in\Gamma, s_k\in\mathcal{S}}\frac{q^{v_j}_{\gamma_i,s_k}}{\widetilde{\lambda}d(v_j)} \le (1+\varsigma)\right] \ge 1 - exp(-\chi n) \tag{14}$$

we can prove Lemma 5 in a similar way to the proof of Lemma 3. Thus, we omit the detailed proof here.

Based on the above analyses, we can draw two conclusions: First, we propose a polynomial time algorithm which achieves an $(O(\frac{\log n}{\varpi}+1), O(\frac{\log n}{\varpi}+1))$ approximation ratio for the general traffic pattern. Second, under the assumption of $\varpi = \Omega(n)$, which is also common in practice, the algorithm approaches near-optimal performance for large-scale networks with a high probability (almost close to 1).

IV. Online Algorithm for RSC

In this section, we will design an online algorithm for flow routing and measurement with link capacity and switch computing capacity constraints. The online scenario is based on the following assumptions. First, we do not know which

macroflows will arrive at the network, and assume that the macroflow requests arrive one-by-one. Second, when a macroflow γ_i arrives, we can estimate its traffic size $f(\gamma_i)$ and the number of packets $N(\gamma_i)$ by the historical traffic statistics information, as described in Section II-C. Finally, we assume each macroflow will never leave once it arrives for simplifying the performance analyses. To be more practical, we extend our algorithm to the general case where macroflows dynamically arrive and leave (Section IV-C). Under the offline scenario, the goal of RSC is to achieve the load balancing based on long-term traffic observation. Different from the offline scenario, we use network throughput as the performance metric of the online algorithm. Since the resources (*e.g.*, link capacity) are limited, not all flows will be served by the network.

A. Algorithm Description

Due to the immediacy of macroflows, the controller needs to response the new-arrival macroflows quickly. For each arrival macroflow, there have constructed multiple feasible paths, each of which may associate with multiple available sketch configurations for traffic measurement. For simplicity, we use h to represent a routing and measurement (RM) scheme including one routing path and one sketch configuration, which can be denoted as {routing path, sketch configuration}. For example, there are three feasible paths $\{p_1, p_2, p_3\}$ for macroflow γ_i. We assume there are two switches $\{v_1, v_2\}$ on routing path p_1 and three sketches, called $\{s_1, s_2, s_3\}$, are deployed on both v_1 and v_2. We can conclude that there are totally 8 different sketch configurations. For path p_1, the sketch configurations can be listed as $[(s_1, v_1), (s_2, v_1), (s_3, v_1)]$, $[(s_1, v_2), (s_2, v_1), (s_3, v_2)]$, …and $[(s_1, v_2), (s_2, v_2), (s_3, v_2)]$. Without confusion, the RM scheme $\{p_1, [(s_1, v_1), (s_2, v_1), (s_3, v_2)]\}$ means the macroflow γ_i will be routed on path p_1 and be measured by sketches s_1, s_2 on switch v_1 and sketch s_3 on switch v_2. Likewise, for paths p_2 and p_3, assume they have 12 and 8 sketch configurations, respectively. For each macroflow $\gamma_i \in \Gamma$, we use set $\mathcal{H}_{\gamma_i}$ to include all its RM schemes. Then, $|\mathcal{H}_{\gamma_i}| = 8 + 12 + 8 = 28$. We will choose one RM scheme for each macroflow so that the network throughput can be maximized without network congestion.

Accordingly, online RSC can be formalized as follows:

$$\max \sum_{\gamma_i \in \Gamma} \sum_{h \in \mathcal{H}_{\gamma_i}} z_{\gamma_i}^h \cdot f(\gamma_i)$$

$$S.t. \begin{cases} \sum\limits_{h \in \mathcal{H}_{\gamma_i}} z_{\gamma_i}^h \le 1, & \forall \gamma_i \in \Gamma \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{e \in h : h \in \mathcal{H}_{\gamma_i}} z_{\gamma_i}^h \cdot f(\gamma_i) \le c(e), & \forall e \in E \\ \sum\limits_{\gamma_i \in \Gamma(v_j, s_k)} \sum\limits_{h : h \in \mathcal{H}_{\gamma_i}} z_{\gamma_i}^h \cdot t(s_k) N(\gamma_i) \le d(v_j), & \forall v_j \in V \\ \sum\limits_{\gamma_i \in \Gamma} \sum\limits_{v_j \in h : h \in \mathcal{H}_{\gamma_i}} z_{\gamma_i}^h \le s(v_j), & \forall v_j \in V \\ z_{\gamma_i}^h \in \{0, 1\}, & \forall h, \gamma_i \end{cases} \tag{15}$$

The first set of inequalities expresses the controller will choose an RM scheme for each macroflow. The second, third and fourth sets of inequalities represent the link capacity constraints, the computing capacity constraints, and the rule-table threshold constraints, respectively.

To solve the problem in Eq. (15), we design an online algorithm based on the primal-dual, called PD-RSC. We consider the dual problem for the linear relaxation of Eq. (15):

$$\min \sum_{\gamma_i \in \Gamma} \theta(\gamma_i) + \sum_{e \in E} \alpha(e) c(e) + \sum_{v_j \in V} (\beta(v_j) d(v_j) + \pi(v_j) s(v_j))$$

$$S.t. \begin{cases} \theta(\gamma_i) \ge f(\gamma_i) - \sum\limits_{e \in E} \delta(\gamma_i, h, e) \cdot f(\gamma_i) \cdot \alpha(e) \\ \quad - \sum\limits_{v_j \in V} \sum\limits_{s_k \in S} \delta(\gamma_i, h, v_j, s_k) \cdot t(s_k) \cdot N(\gamma_i) \cdot \beta(v_j) \\ \quad - \sum\limits_{v_j \in V} \delta(\gamma_i, h, v_j) \cdot \pi(v_j), & \forall \gamma_i \in \Gamma,\ h \in \mathcal{H}_{\gamma_i} \\ \theta(\gamma_i) \ge 0, & \forall \gamma_i \in \Gamma \\ \alpha(e) \ge 0, & \forall e \in E \\ \beta(v_j) \ge 0, & \forall v_j \in V \\ \pi(v_j) \ge 0, & \forall v_j \in V \end{cases} \tag{16}$$

Note that $\theta(\gamma_i)$, $\alpha(e)$, $\beta(v_j)$ and $\pi(v_j)$ are the dual variables of the first set to the fourth set of inequalities in Eq. (15). Meanwhile, these dual variables should be non-negative. The function δ expresses the occupation of each category of resources by a macroflow. More specifically, the functions $\delta(\gamma_i, h, e)$ and $\delta(\gamma_i, h, v_j)$ denote whether link e and switch v_j lie on the routing path of scheme h for macroflow γ_i or not. $\delta(\gamma_i, h, v_j, s_k)$ denotes whether sketch s_k on switch v_j will measure macroflow γ_i or not if scheme h is chosen for this macroflow.

$$\begin{cases} \delta(\gamma_i, h, e) = \begin{cases} 1, & \text{if } \gamma_i \text{ routes through link } e \text{ using } h. \\ 0, & otherwise. \end{cases} \\ \delta(\gamma_i, h, v_j) \\ = \begin{cases} 1, & \text{if } \gamma_i \text{ routes through switch } v_j \text{ using } h. \\ 0, & otherwise. \end{cases} \\ \delta(\gamma_i, h, v_j, s_k) \\ = \begin{cases} 1, & \text{if } \gamma_i \text{ is measured by sketch } s_k \\ & \text{on switch } v_j \text{ using } h. \\ 0, & otherwise. \end{cases} \end{cases} \tag{17}$$

The PD-RSC algorithm is formally described in Alg. 2. The first step of PD-RSC is to initialize corresponding constants and dual variables. According to the first set of inequalities in Eq. (16), we define B^* as the maximum usage of each resource over all possible RM schemes of all macroflows.

$$B^* = \max_{\gamma_i, h} \{ \max_e \delta(\gamma_i, h, e), \max_{v_j} \frac{\delta(\gamma_i, h, v_j)}{f(\gamma_i)}, \max_{v_j} \frac{\sum\limits_{s_k \in S} \delta(\gamma_i, h, v_j, s_k) \cdot t(s_k) N(\gamma_i)}{f(\gamma_i)} \} \tag{18}$$

The constant $\mathcal{N}$ is the number of inequalities from the second set to the fourth set in Eq. (15). That is, $\mathcal{N} = 2|V| + |E|$. The constant $\epsilon \in [0, 1]$ denotes a trade-off between the resource violation and network throughput. All the dual variables $\theta(\gamma_i)$, $\alpha(e)$, $\beta(v_j)$ and $\pi(v_j)$ are initialized to 0.

The second step is to choose an RM scheme for each arrival macroflow. When macroflow γ_i arrives, we compute the profit of each RM scheme $h \in \mathcal{H}_{\gamma_i}$, which represents the utility over all types of resources for macroflow γ_i. The profit of scheme

Algorithm 2 PD-RSC: Online Primal-Dual Algorithm for RSC

Step 1: Algorithm Initialization
Initialize constants B^*, $\mathcal{N}$ and $\epsilon \in (0,1)$
Initialize variables $\alpha(e)=0, \forall e \in E$ $\beta(v_j)=0, \forall v_j \in V$ $\pi(v_j)=0, \forall v_j \in V$ $\theta(\gamma_i)=0, \forall \gamma_i \in \Gamma$
Step 2: On Arriving of a Macroflow γ_i
for each $h \in \mathcal{H}_{\gamma_i}$ **do**
  Compute K_h by Eq. (19)
$K \rightarrow \max_{h \in \mathcal{H}_{\gamma_i}} K_h$
if $K < 0$ **then**
  Reject the macroflow γ_i
else
  $h^* \rightarrow \arg\max_{h \in \mathcal{H}_{\gamma_i}} K_h$
  Route and measure the macroflow according to scheme h^*
  Update $\theta(\gamma_i)$ as K
  Update $\alpha(e)$, $\pi(v_j)$ and $\beta(v_j)$ by Eqs. (20), (21) and (22).

h is denoted as K_h:

$$K_h = f(\gamma_i) - \sum_{v_j \in V} \sum_{s_k \in S} \delta(\gamma_i, h, v_j, s_k) \cdot t(s_k) N(\gamma_i) \beta(v_j) - \sum_{e \in E} \delta(\gamma_i, h, e) \cdot f(\gamma_i) \alpha(e) - \sum_{v_j \in V} \delta(\gamma_i, h, v_j) \cdot \pi(v_j) \quad (19)$$

Then we determine the maximum profit of all possible RM schemes for macroflow γ_i, that is, $K = max_{h \in \mathcal{H}_{\gamma_i}} K_h$. If K is negative, we believe that the forwarding of this macroflow does not bring any profit. So, this macroflow will be rejected. Otherwise, we will choose an efficient RM scheme for macroflow γ_i. We use h^* to represent the RM scheme with the maximum profit K, that is, $h^* \rightarrow \arg\max_{h \in \mathcal{H}_{\gamma_i}} K_h$. Macroflow γ_i will be routed and measured according to the RM scheme h^*. Then, we set the dual variable $\theta(\gamma_i)$ of macroflow γ_i as the maximum profit K. Meanwhile, since the available resources of links and switches on the routing path of this macroflow are gradually decreased, the dual variables of the occupied resources will increase accordingly. To avoid the confusion, we use $\alpha(e)$ and $\alpha'(e)$ to denote the dual variables before and after resource occupation. The dual variable update of the link resource is as follows:

$$\alpha'(e) = \alpha(e) \left[1 + \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{c(e)}\right] + \epsilon \cdot \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{\mathcal{N} \cdot c(e) B^*}, \quad \forall e \in E \quad (20)$$

For the flow table resource and switch computing resource, the dual variable updates are as follows:

$$\pi'(v_j) = \pi(v_j) \left[1 + \frac{\delta(\gamma_i, h^*, v_j)}{s(v_j)}\right] + \epsilon \cdot \frac{\delta(\gamma_i, h^*, v_j)}{\mathcal{N} \cdot s(v_j) B^*}, \quad \forall v_j \in V \quad (21)$$

$$\beta'(v_j) = \beta(v_j) \prod_{s_k \in S} \{ \left[1 + \frac{\delta(\gamma_i, h^*, v_j, s_k) \cdot t(s_k) N(\gamma_i)}{d(v_j)}\right] + \epsilon \cdot \frac{\delta(\gamma_i, h^*, v_j, s_k) \cdot t(s_k) N(\gamma_i)}{\mathcal{N} \cdot d(v_j) B^*} \}, \quad \forall v_j \in V \quad (22)$$

In Eq. (21) and (22), $\pi'(v_j)$ and $\beta'(v_j)$ denote the values of $\pi(v_j)$ and $\beta(v_j)$ after resource occupation, respectively.

B. Performance Analysis

In this section, we analyse the competitive performance of PD-RSC.

Definition 4: An online algorithm for RSC is $[\kappa, \psi]$ competitive if it achieves at least $\kappa \cdot$ OPT, where OPT is the optimal network throughput for RSC, and constraints are violated by at most a multiplicative factor ψ [45].

For the utility competitiveness factor κ and the violation level ψ, the former means how much we lose by having restricted access to flow information under the online scenario, and the latter measures how much the algorithm will overload on some resources. In the best case, the online algorithm achieves $[1, 0]$ competitive, or at least $[\varrho, 0]$ competitive for some $\varrho > 0$. However, an online algorithm with a positive competitive ratio must violate constraints. According to [45], it is possible to construct a special traffic distribution that makes ψ close to $\Omega(log\mathcal{N})$ for all online algorithms. It also means that if we insist on non-negative competitive performance, the best in terms of resource overloading is a logarithmic bound.

Theorem 6: PD-RSC is $[(1-\epsilon), (log\mathcal{N} + log(1/\epsilon))]$ competitive.

The detailed proof has been analyzed in Appendix. This result tells us that the online PD-RSC algorithm reaches a theoretical lower bound in terms of resource overloading.

Based on the above analysis, our primal-dual online algorithm can achieve a $[(1-\epsilon), (log\mathcal{N} + log(1/\epsilon))]$ competitive for all traffic pattern. This also means the performance of PD-RSC is superior even if the traffic pattern is adversarial.

C. Discussion

Under the online scenario, we assume that one macroflow will not terminate. We discuss the macroflow termination scenario as a supplement to the previous assumption. To detect the termination of a macroflow, we adopt the flow entry timeout mechanism [13]. The controller can set a valid time (*e.g.*, 500ms) when it installs a flow entry on a switch. If a flow entry does not match any packets for the pre-set time, the switch regards that the macroflow matching the flow entry has been terminated and actively removes the flow entry. Then the controller receives a "Flow-Mod" message from the switch. After a macroflow terminates, some resources occupied by this macroflow will be released, and the dual variables of the released resources will be decreased accordingly. The reduction of dual variables is similar to Eqs. (20)-(22) in Section IV-A. We take the link resource as an example. To avoid the confusion, we use $\alpha(e)$ and $\alpha''(e)$ to denote the dual variables before and after resource releasing. Similar to Eq. (20), we have

$$\alpha(e) = \alpha''(e) \left[1 + \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{c(e)}\right] + \epsilon \cdot \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{\mathcal{N} \cdot c(e) B^*}, \quad \forall e \in E \quad (23)$$

We transform Eq. (23) into

$$\alpha''(e) = \left[\alpha(e) - \epsilon \cdot \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{\mathcal{N} \cdot c(e) B^*}\right] / \left[1 + \frac{\delta(\gamma_i, h^*, e) f(\gamma_i)}{c(e)}\right], \quad \forall e \in E \quad (24)$$

Similarly, the updates of variables $\pi(v_j)$ and $\beta(v_j)$ are as follows.

$$\pi''(v_j) = \left[\pi(v_j) - \epsilon \cdot \frac{\delta(\gamma_i, h^*, v_j)}{\mathcal{N} \cdot s(v_j) B^*}\right] / \left[1 + \frac{\delta(\gamma_i, h^*, v_j)}{s(v_j)}\right], \quad \forall v_j \in V \quad (25)$$

$$\beta''(v_j) = \beta(v_j) \prod_{s_k \in S} \{ \left[1 - \epsilon \cdot \frac{\delta(\gamma_i, h^*, v_j, s_k) \cdot t(s_k) N(\gamma_i)}{\mathcal{N} \cdot d(v_j) B^* \cdot \beta(v_j)} \right] / \left[1 + \frac{\delta(\gamma_i, h^*, v_j, s_k) \cdot t(s_k) N(\gamma_i)}{d(v_j)} \right] \}, \quad \forall v_j \in V \tag{26}$$

Those key notations in Eqs. (24)-(26) can be found in Section IV-A. The generalized online algorithm, called GPD-RSC, is formally described in Alg. 3. The former two steps of GPD-RSC are the same as those of PD-RSC. The third step is to release the occupied network resources after a macroflow terminates. For simplicity, the routing and measurement scheme of macroflow γ_i is denoted as h^*. When macroflow γ_i terminates, the algorithm sets the variable $\theta(\gamma_i)$ to zero, and updates the variables $\alpha(e)$, $\pi(v_j)$ and $\beta(v_j)$ by Eqs. (24), (25) and (26), respectively.

Algorithm 3 GPD-RSC: Generalized Online Algorithm for RSC

1: **Step 1: the same as that in PD-RSC**
2: **Step 2: the same as that in PD-RSC**
3: **Step 3: On termination of macroflow** γ_i
4: Let h^* denote the routing and measurement scheme for γ_i
5: Set $\theta(\gamma_i)$ as 0
6: Update $\alpha(e)$, $\pi(v_j)$ and $\beta(v_j)$ by Eqs. (24), (25) and (26).

V. Dealing With Traffic Burstiness

Network dynamics may result in traffic burstiness, which will negatively affect network performance and user experience. On an SDN switch, its forwarding rate is usually much faster than its measurement rate. When traffic burst occurs on a switch, though all the traffic may be successfully forwarded, traffic measurement may be blocked. To avoid the measurement congestion, each switch maintains a buffer with a given length L (*e.g.*, 500KB). We design an adaptive queueing-theory-based packet sampling (QTPS) algorithm for high-fidelity traffic measurement.

A. Algorithm Description

In this section, the QTPS algorithm is proposed to sample the arrival packets for traffic measurement with congestion avoidance. For ease of analysis, we assume the packet arrival obeys a Poisson distribution with parameter ϕ. Based on the properties of Poisson distribution, we have $P(X = k) = \frac{\phi^k \exp(-\phi)}{k!}$ and $E(X) = \phi$, where $E(X)$ represents the expected number of arrival packets per unit time. We estimate the packet-arrival rate ϕ by the number of arrival packets through a time slot. Moreover, since the sketch measurement is independent for each sampled packet, we assume that sketch measurement obeys a negative exponential distribution with parameter μ (*e.g.*, 10Mpps). μ denotes the traffic measurement rate on this switch, which can be obtained through long-time observation. Based on the above assumptions, the process of packet arrival and measurement can be modeled as an $M/M/1/N$ queue system [46].

QTPS is described in Alg. 4. In the first step, QTPS will initialize some parameters for determining the sampling probability. The packet head will enter the buffer with a sampling probability p, and be dropped in case that the buffer is full. $\tau \in (0, 1)$ denotes the serving ratio, which is the expected ratio between the number of measured packets and the number of sampled packets. For example, if τ is 99%, we expect that only one packet will be dropped per one hundred sampled packets on average. σ is a threshold to adjust the sampling probability for measurement congestion avoidance, and satisfies the inequality in Line 4 of Alg. 4. The second step is to adjust the sampling probability with traffic dynamics. For each time slot t, we derive the sum of the packet statistics of the ingress ports on a switch as $count_1$, and use $count_2$ to denote the number of arrival packets before this time slot. Then we estimate the packet arrival rate ϕ as the difference between $count_1$ and $count_2$ divided by t. In case of $\phi > \sigma \cdot \mu$, we believe that there may occur traffic burst, and should dynamically adjust the sampling probability $p = \frac{\sigma \cdot \mu}{\phi}$. Otherwise, there is no traffic burst and the sampling probability p is set to 1. Note that the count of the sketch is increased by $1/p$ every sampled packet when p is less than 1. At the end of each time slot t, we update the value of $count_2$.

Algorithm 4 QTPS: Queueing-Theory-Based Packet Sampling

1: **Step 1: Algorithm Initialization**
2: Initialize packet measurement rate μ, $\tau \in (0, 1)$
3: Set sampling probability $p = 1$, $count_1 = 0$ and $count_2 = 0$
4: Derive $\sigma \in (0, 1)$ satisfies $L \cdot ln\sigma + ln(1 - \tau \cdot \sigma) \leq ln(1 - \tau)$
5: **Step 2: Updating Sampling Probability**
6: **for** each time slot t **do**
7: Derive the sum of the packet statistics of the ingress ports as $count_1$
8: $\phi = \frac{count_1 - count_2}{t}$
9: **if** $\phi \leq \sigma \cdot \mu$ **then**
10: $p = 1$
11: **else**
12: $p = \frac{\sigma \cdot \mu}{\phi}$
13: $count_2 = count_1$

B. Performance Analysis

In this section, we analyse the performance of QTPS.

Theorem 7: Without bursty traffic or $p = 1$, the QTPS algorithm guarantees that the actual serving ratio $\bar{\tau}$ is not less than τ.

Proof: According to the property of an $M/M/1/N$ queue system, we obtain the balance equations as follows:

$$\begin{cases} \phi \cdot P_0 = \mu \cdot P_1, & \\ \phi \cdot P_{n-1} + \mu \cdot P_{n+1} = \phi \cdot P_n + \mu \cdot P_n, & n < L \\ \mu \cdot P_n = \phi \cdot P_{n-1}, & n \leq L \end{cases} \tag{27}$$

P_n denotes the probability that there are n units of packet information in the buffer. Therefore,

$$P_0 + P_1 + \ldots + P_L = 1 \tag{28}$$

Combining Eqs. (27) and (28), we have

$$\begin{cases} P_0 = \dfrac{1 - \dfrac{\phi}{\mu}}{1 - (\dfrac{\phi}{\mu})^{L+1}} \\ P_n = P_0 \cdot (\dfrac{\phi}{\mu})^n \end{cases} \tag{29}$$

By line 4 of Alg. 4, we get the following inference:

$$\begin{aligned} & L \cdot ln\sigma + ln(1 - \tau \cdot \sigma) \leq ln(1 - \tau) \\ & \Rightarrow \sigma^L \cdot (1 - \tau \cdot \sigma) \leq 1 - \tau \\ & \Rightarrow \tau \leq \frac{1 - \sigma^L}{1 - \sigma^{L+1}} \end{aligned} \tag{30}$$

Combining Eqs. (29) and (30), we get the actual serving ratio $\bar{\tau}$ as follows:

$$\begin{aligned}\bar{\tau} &= 1 - P_L = 1 - P_0 \cdot (\frac{\phi}{\mu})^L \\ &= 1 - \frac{1 - \frac{\phi}{\mu}}{1 - (\frac{\phi}{\mu})^{L+1}} \cdot (\frac{\phi}{\mu})^L \\ &\geq \frac{1 - \sigma^L}{1 - \sigma^{L+1}} \geq \tau \end{aligned} \tag{31}$$

□

Theorem 8: In the case of traffic burstiness or $p < 1$, QTPS guarantees that the actual serving ratio $\bar{\tau}$ is not less than τ too.

Proof: We first prove that the sampled traffic with probability p still satisfies a Poisson distribution with parameter $p \cdot \phi$. The probability of sampling k packets from n packets with probability p is:

$$B(n,k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{32}$$

Then the probability of sampling k packets is

$$\begin{aligned} &P(Y = k) \\ &= \sum_{n=k}^{\infty} B(n,k) \cdot P(X = n) \\ &= \sum_{n=k}^{\infty} \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \cdot \frac{\phi^k \exp(-\phi)}{k!} \\ &= \frac{1}{k!} \cdot (\frac{p}{1-p})^k \cdot \exp(-\phi) \cdot \sum_{n=k}^{\infty} \frac{(\phi(1-p))^n}{(n-k)!} \\ &= \frac{(\phi(1-p))^k}{k!} (\frac{p}{1-p})^k \cdot \exp(-\phi) \cdot \sum_{t=0}^{\infty} \frac{(\phi(1-p))^t}{t!} \\ &= \frac{(\phi \cdot p)^k}{k!} \cdot \exp(-\phi \cdot p) \end{aligned} \tag{33}$$

Eq. (33) reflects that the packets sampled with probability p satisfy a Poisson distribution of parameter $\phi \cdot p$. According to Theorem 7, we can get the same conclusion. □

Based on the above analysis, we can draw a conclusion. Nearly all sampled packets can be measured by the way of setting the proper value of τ. In other words, QTPS solves the measurement congestion that may be caused by traffic burstiness.

VI. Performance Evaluation

A. Simulation Settings

We perform our simulations over two types of topologies, the fat-tree topology [47] and the Monash campus topology [48]. The former is commonly adopted in data centers and includes 80 switches (32 edge switches, 32 aggregation switches, and 16 core switches) and 192 servers. The latter is a typical campus network topology and contains 100 switches and 200 hosts. The two topologies represent networks with different features. Specifically, the fat-tree data center topology is a symmetrical network while the Monash campus topology is asymmetrical. For the above two topologies, we set the link capacity as 10Gbps and the switch's CPU processing ability for network measurement as 3.0GHz.

In the simulations, we generate flows with two types of size distribution: (1) the 2-8 distribution, where 20% of all flows account for 80% of traffic volume [15]. For this distribution, the average traffic intensity of mice flows and elephant flows is set as 0.25Mbps and 4Mbps, respectively. (2) The Gaussian distribution [49], in which the average flow intensity is set as 1Mbps. The packet size is set as 1KB uniformly. We generate two types of flows to simulate the practical traffic: (1) random flows, whose sources and destinations are randomly choosed; (2) server flows, which simulate the traffic between random hosts and multiple designated servers, *e.g.*, web servers [50], [51]. To support multiple measurement tasks together, we deploy five sketches, Flowradar [19], Twolevel [25], MRAC [26], k-Min [52] and Count-Min [20], respectively, on each SDN switch. These sketches can support various measurement tasks: 1) Flowradar can detect the heavy hitter and heavy changer in flow traffic; 2) Twolevel can detect the DDos and Superspreader attack in the network; 3) Flow size distribution estimation and cardinality estimation can be supported by MRAC and k-Min, respectively; 4) Count-Min enables fast measurement for per-flow frequency. The average computing overheads per packet of these sketches are 2584, 4292, 404, 2388 and 78 CPU cycles [23], respectively. For each switch, we only consider the computing overhead by sketches and ignore other overheads, such as that of forwarding packets.

The simulations are performed under three network scenarios. The first scenario is an offline scenario, in which we know all the traffic information. The second scenario is an online scenario, in which we only know the current traffic information. For both the offline and online scenarios, the simulations are performed using 20×10^4 flows. The final one is the bursty traffic scenario. We evaluate the packet loss ratio and the relative measurement error over 6-second bursty traffic from our campus network. In the last scenario, we set the memory size of the queue as 500KB, and the expected serving ratio as 99%. Compared with the SRAM size on switches, 500KB is relatively small. Since the size of one TCP or IP packet head is about 20B, the queue can record the head information of 2.5×10^4 packets at most. The switch updates the sampling probability p 10 times every second. For the above scenarios, we run each simulation 20 times and calculate the average value as the simulation results.

B. Performance Metrics and Benchmarks

In our simulations, we use the following five performance metrics: (1) the link load ratio; (2) the switch overhead ratio; (3) the network throughput; (4) the average relative measurement error; and (5) the relative error CDF for all large flows. For each simulation instance, we measure the traffic load of each link and the computing overhead of each switch. For each link, its load ratio is its load divided by the link capacity, and we use their largest value as the first metric. The overhead ratio is the switch computing overhead divided by its computing capacity, and we use their largest value as the second metric. Moreover, the load balancing factor λ is determined by the maximum value of the first and the second metrics. We also measure the throughput as the third metric. The latter two metrics are for the bursty traffic scenario. We compute the relative measurement error of each large flow and average the result as the fourth metric. The last metric is the cumulative distribution function (CDF) of the relative measurement error for all large flows.

We choose four benchmarks for performance comparison. The first benchmark is the optimal result, denoted as OPT-LP, for the offline RSC problem, which can be derived by optimally solving the linear programming of Eq. (2). The second benchmark represents a category solution that separately

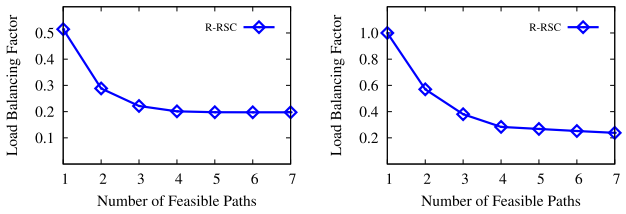


Fig. 3. Load balancing factor vs. Number of feasible paths. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

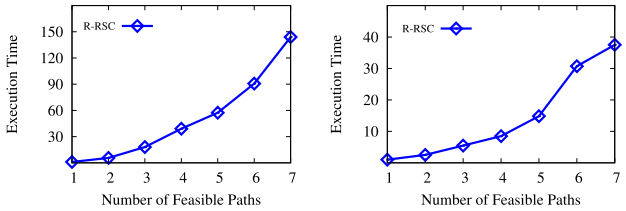


Fig. 4. Execution time vs. Number of feasible paths. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

considers the routing selection and sketch configuration. When a flow arrives, the controller will choose one path using the MCF method [51]. After the routing path is chosen, the sketch configuration is implemented using a greedy principle. That is, the switch with more computing resources on the routing path will configure a sketch, until this flow is measured by all these sketches. We call this benchmark as MCF-Greedy. The third one, called Joint-Greedy, represents a category solution that considers the joint route selection and sketch configuration. The joint approach follows the greedy principle. That is, the controller chooses the routing path from those equal-paths and the measurement scheme with the minimum impact on load balancing factor for each flow. For the QTPS algorithm, we choose a benchmark, called without sampling (WSP), which will directly drop the packets without sampling when the queue is full.

C. Simulation Results

Here we run four sets of simulations for performance evaluation. The first set of simulations observes the impact of the number of feasible paths on load balancing, in which 5×10^4 flows are generated in the network. In Fig. 3, as the number of feasible paths per flow increases, the network will gradually become more load balancing. When there are more than five feasible paths for each flow, the load balancing performance almost remains. However, the execution time increases significantly with more feasible paths for a flow, as shown in Fig. 4, in which the execution time of R-RSC under one feasible path per flow is set as 1. It is not difficult to make a trade-off optimization between the number of feasible paths and the execution time. By these two figures, we explore five feasible paths for each flow in the following simulations.

The second set of simulations compares the link load ratio and the switch overhead ratio for R-RSC with three benchmarks: OPT-LP, MCF-Greedy and Joint-Greedy. The simulation results under 2-8 distribution and Gaussian distribution are shown in Figs. 5-6 and Figs. 7-8, respectively. As the number of flows increases, the link load ratio and the switch overhead ratio accordingly increase. However, the increasing rate of R-RSC is much slower than that of both MCF-Greedy and Joint-Greedy. More specifically, we observe that R-RSC exhibits very close load balancing performance with the optimal solution under various flow size distributions. This shows

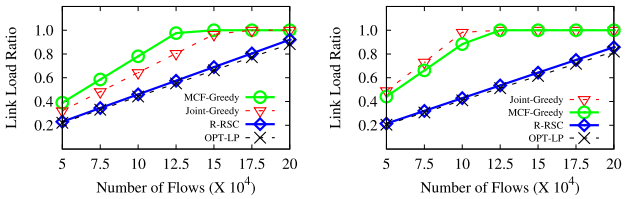


Fig. 5. Link load ratio vs. number of flows with 2-8 distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

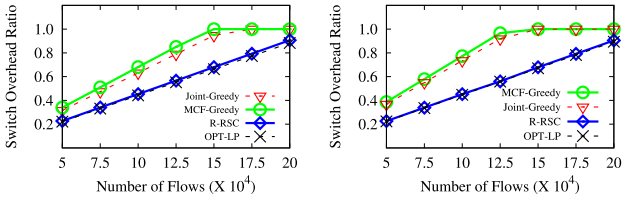


Fig. 6. Switch overhead ratio vs. Number of flows with 2-8 distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

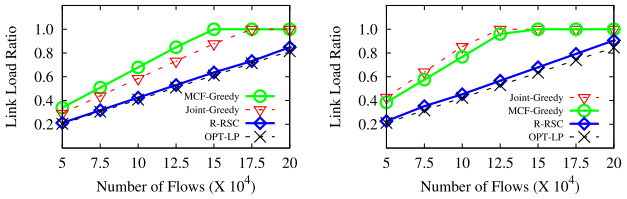


Fig. 7. Link load ratio vs. Number of flows with Gaussian distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

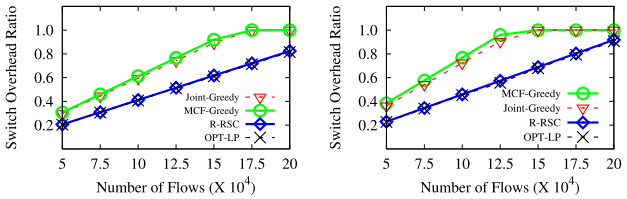


Fig. 8. Switch overhead ratio vs. Number of flows with Gaussian distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

that R-RSC can achieve near-optimal performance. From the left plots of Figs. 5-6, when there are 10×10^4 flows in the network, the link load ratio and the switch overhead ratio are 0.45 and 0.42 in the Monash topology by R-RSC, respectively. The load balancing factor λ is $\max\{0.45, 0.42\} = 0.45$. For the MCF-Greedy solution, the load balancing factor is $\max\{0.78, 0.68\} = 0.78$. For Joint-Greedy, the link load ratio and the switch overhead ratio are 0.64 and 0.63, respectively. The load balancing factor is $\max\{0.64,\ 0.63\} = 0.64$. This shows that R-RSC can achieve better load balancing compared with two benchmarks. Moreover, as shown in Figs. 7-8, when the number of flows increases, we observe that switches and links have different degrees of overload by the two benchmarks. Based on the simulation results, R-RSC achieves an average of 40% performance improvement comparing with MCF-Greedy. That is because we combine the sketch configuration with flow routing as a whole, which plays an important role in network load balancing.

Our third set of simulations indicates the network throughput of the PD-RSC algorithm. We generate 20×10^4 flows in the simulations. Figs. 9-10 show the simulation results under different flow size distributions. When there is relatively less traffic in the network, the performance of all algorithms is

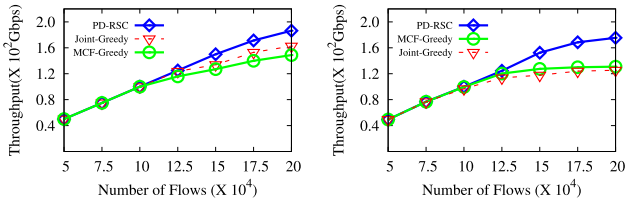


Fig. 9. Network throughput vs. Number of flows with 2-8 distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

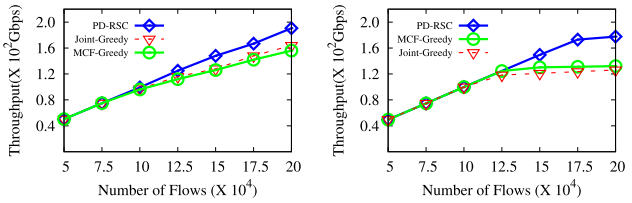


Fig. 10. Network throughput vs. Number of flows with Gaussian distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

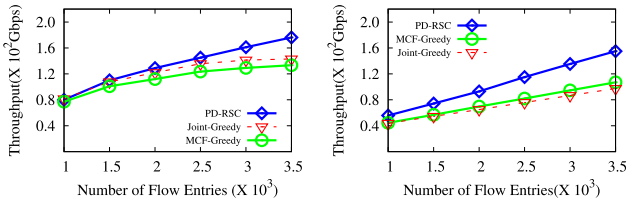


Fig. 11. Network throughput vs. Number of flow entries with 2-8 distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

roughly the same. However, as the flows arrive, the advantage of PD-RSC becomes clear. For example, by the left plot of Fig. 9, in the case of 20×10^4 flows in the network, the PD-RSC, MCF-Greedy, and Joint-Greedy algorithms achieve the network throughput of 182.6Gbps, 145.5Gbps, 161.2Gbps, respectively. From the right plot of Fig. 10, when there are more than 15×10^4 flows, the throughput of two benchmarks is no longer increasing. This shows that PD-RSC can significantly improve the network throughput compared with two benchmarks. For each new-arrival flow, PD-RSC chooses the optimal RM scheme with the maximum profit. After the flow is accepted, if the resources are occupied, the dual variables of occupied resources are also increased. This mechanism helps balance the use of network resources and reduce the probability of network congestion. We also evaluate how the number of updated flow entries on each switch affects the performance of PD-RSC on two topologies, as shown in Figs. 11-12. As the number of updated flow entries on switches increases, the network throughput of PD-RSC increases faster than that of two benchmarks. This shows that PD-RSC employs the flow table resources efficiently comparing with two benchmarks. Comparing with two benchmarks, PD-RSC improves the network throughput about 30% on average.

Our last set of simulations evaluates the average measurement error by QTPS over real network traffic traces. Under this scenario, we assume that the packets will be processed on a Count-Min sketch [20] with the measurement speed of 10M packets per second. In Fig. 13, we observe the change process of sampling probability with the traffic arrival rate. When the traffic arrival rate does not exceed the sketch measurement rate, the sampling probability is set to 1. When traffic burst

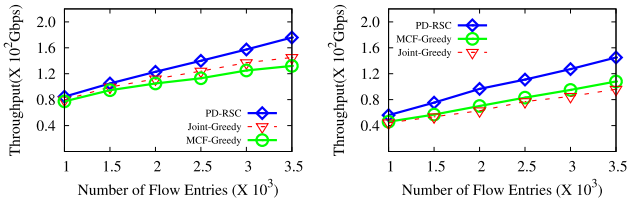


Fig. 12. Network throughput vs. Number of flow entries with Gaussian distribution. *Left Plot*: Monash topology; *Right Plot*: fat-tree topology.

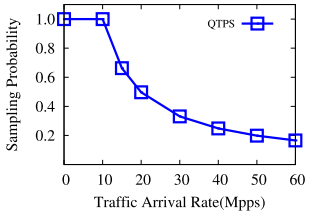


Fig. 13. Sampling probability vs. Traffic arrival rate.

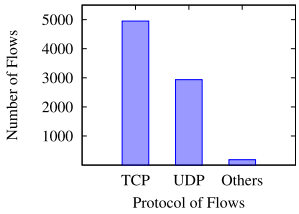


Fig. 14. Number of flows vs. Protocol of flows.

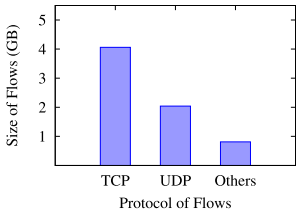


Fig. 15. Size of flows vs. Protocol of Flows.

occurs, QTPS will dynamically adjust the sampling probability to prevent congestion in the packet head queue. In Figs. 14-15, the simulations run on 6-second bursty traffic from our campus network. The bursty traffic consists of 4950 TCP flows, 2937 UDP flows, and 185 flows of other protocols (*e.g.*, SMTP, HTTP). Their traffic size is 4.06GB, 2.04GB, and 0.81GB, respectively. Since the Count-Min sketch will lead to a large error for measuring small flows, we only consider the average error of the maximum ten (Top-10) flows and those large (exceeds 10MB) flows in the traffic. For measurement of the Top-10 flows, QTPS helps to achieve less and more stable measurement error, as shown in Fig. 16. The measurement error of the Top-10 flows by QTPS is less than 8%. However, using WSP, the highest error ratio is up to 50%. In Fig. 17, the relative measurement error of large flows is around 5% and 30% by QTPS and WSP, respectively. The simulation results show that the measurement error by WSP is six times as that of QTPS. That is because bursty traffic will cause congestion in the queue and result in the loss of a large number of packets by WSP. However, QTPS uses a probabilistic sampling mechanism to avoid congestion for bursty traffic. Moreover, we compare the average measurement error and compute its CDF for both QTPS and WSP, as shown in Fig. 18. For QTPS, there are about 80% of the large flows whose measurement

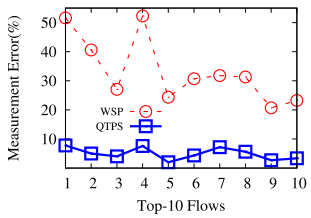


Fig. 16. Measurement error vs. Top-10 flows.

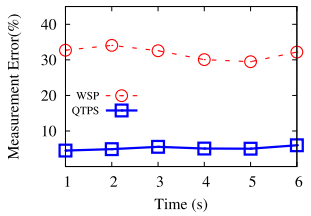


Fig. 17. Average relative measurement error vs. Time.

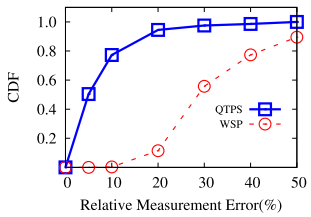


Fig. 18. CDF vs. Relative measurement error.

error does not exceed 10%. In fact, the measurement error is mostly between 20%-50% by WSP. The measurement error by QTPS consists of two parts: (1) the error due to hash collisions; (2) the error caused by packet sampling. The first part is unavoidable unless we increase the memory size for the sketch. For the second part, QTPS can measure all packets without any packet loss when flows arrive quitely, and it can also reduce the loss caused by traffic burst.

According to the simulation results, we can draw three conclusions. First, by Figs. 5-8, R-RSC can achieve the load balancing under different flow size distributions on both two network topologies. Moreover, R-RSC achieves an average of 30% performance improvement comparing with MCF-Greedy. Second, Figs. 9-12 show that PD-RSC improves the network throughput at least 30% comparing with two benchmarks. Third, by Figs. 16-18, the QTPS algorithm successfully avoids measurement congestion and the measurement error of those large flows is only 1/6 as that of WSP.

VII. Conclusion

In this paper, we have studied the joint flow routing and sketch configuration for packet forwarding and measurement simultaneously. We formulate the joint optimization of routing and sketch configuration as an NP-hard problem. To solve this problem, we have designed a rounding-based offline algorithm and a primal-dual-based online algorithm. Their approximation/competitive performance is also analyzed. To deal with the case of traffic burst, we propose a queue-theory-based packet sampling algorithm for high-fidelity traffic measurement. The extensive simulation results show that our proposed algorithms can achieve superior performance on load balancing, network throughput and measurement error comparing with existing solutions.

Appendix
Proof of Theorem 6

Lemma 9: The update rules in the PD-RSC algorithm preserve the dual feasibility of all the dual variables.

Proof: For dual variables $\alpha(e)$, $\beta(v_j)$ and $\pi(v_j)$, the update always keeps them to be nonnegative, as their value will only increase from zero during the update. For each macroflow γ_i, if it is rejected, we know K_h is negative for $\forall h \in \mathcal{H}_{\gamma_i}$. If the macroflow is accepted, we assign $\theta(\gamma_i)$ as the maximum value of K_h, which is satisfied for all $h \in \mathcal{H}_{\gamma_i}$. Moreover, further updates can only make the right hand of Eq. (19) to be smaller, thus preserving the feasibility of the constraints. □

Lemma 10: Whenever a macroflow γ_i is accepted, the objective value of Eq. (16) will be increased not more than $(1+\epsilon)f(\gamma_i)$.

Proof: By the update rules of the dual variables, when a macroflow γ_i is routed and measured, the objective value of dual problem will be increased not more than

$$\begin{aligned} & f(\gamma_i) + \frac{\epsilon}{\mathcal{N}\cdot B^*}\cdot\sum_{e\in E}\delta(\gamma_i,e)f(\gamma_i) + \frac{\epsilon}{\mathcal{N}\cdot B^*}\cdot\sum_{v_j\in V}\delta(\gamma_i,v_j) \\ & \quad + \frac{\epsilon}{\mathcal{N}\cdot B^*}\cdot\sum_{v_j\in V}\sum_{s_k\in S}\delta(\gamma_i,v_j,s_k)t(s_k)N(\gamma_i) \\ & \le (1+\epsilon\cdot\frac{\mathcal{N}\cdot B^*}{\mathcal{N}\cdot B^*})f(\gamma_i) = (1+\epsilon)f(\gamma_i) \end{aligned} \tag{34}$$

□

For each macroflow $\gamma_i \in \Gamma$ and each link $e \in E$, we use $L(e,\gamma_i)$ and $\alpha(e,\gamma_i)$ to represent the load on link e (after macroflow γ_i has been processed) and the value of $\alpha(e)$, respectively. By the same way, we define the computing overhead as $C(v_j,\gamma_i)$ and the number of matched flow entries as $F(v_j,\gamma_i)$ on switch v_j. We set $\beta(v_j,\gamma_i)$ and $\pi(v_j,\gamma_i)$ as the values of $\beta(v_j)$ and $\pi(v_j)$, respectively.

Lemma 11: For each macroflow γ_i, each link e and each switch v_j, we have

$$\begin{aligned} \alpha(e,\gamma_i) &\ge \epsilon\cdot\frac{exp\left[L(e,\gamma_i)/c(e)\right]-1}{\mathcal{N}\cdot B^*} \\ \beta(v_j,\gamma_i) &\ge \epsilon\cdot\frac{exp\left[C(v_j,\gamma_i)/d(v_j)\right]-1}{\mathcal{N}\cdot B^*} \\ \pi(v_j,\gamma_i) &\ge \epsilon\cdot\frac{exp\left[F(v_j,\gamma_i)/s(v_j)\right]-1}{\mathcal{N}\cdot B^*} \end{aligned} \tag{35}$$

Proof: We prove the first set of inequalities in Eq. (35) by induction of γ_i. In the initial stage of the algorithm, $\alpha(e,\gamma_i) = L(e,\gamma_i) = 0$ for all links e. Thus, the inequalities hold. Then we consider the situation in which a macroflow γ_i arrives. If macroflow γ_i is rejected, the inequality is still satisfied. If it is accepted, we have

$$L(e,\gamma_i) = L(e,\gamma_i-1) + \delta(\gamma_i,h^*,e)f(\gamma_i) \tag{36}$$

and

$$\begin{aligned} \alpha(e,\gamma_i) = \alpha(e,\gamma_i-1)\left[1+\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{c(e)}\right] \\ + \epsilon\cdot\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{\mathcal{N}\cdot B^* c(e)} \end{aligned} \tag{37}$$

By induction hypothesis, it follows

$$\begin{aligned} & \alpha(e,\gamma_i) \\ & \ge \epsilon\cdot\frac{exp\left[\frac{L(e,\gamma_i-1)}{c(e)}\right]-1}{\mathcal{N}\cdot B^*}\left[1+\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{c(e)}\right] \end{aligned}$$

$$
\begin{aligned}
&+\epsilon\cdot\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{\mathcal{N}\cdot B^* c(e)}\\
&=\epsilon\cdot\frac{1}{\mathcal{N}\cdot B^*}\{exp\left[\frac{L(e,\gamma_i-1)}{c(e)}\right]\\
&\quad\times\left[1+\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{c(e)}\right]-1\}\\
&\approx\epsilon\cdot\frac{1}{\mathcal{N}\cdot B^*}\{exp\left[\frac{L(e,\gamma_i-1)}{c(e)}\right]\\
&\quad\times exp\left[\frac{\delta(\gamma_i,h^*,e)f(\gamma_i)}{c(e)}\right]-1\}\\
&=\epsilon\cdot\frac{exp\left[L(e,\gamma_i)/c(e)\right]-1}{\mathcal{N}\cdot B^*}
\end{aligned}
\tag{38}
$$

The proof of the second and the third sets of inequalities in Eq. (35) is similar with the above analysis. We thus omit the proof here. □

Now we prove Theorem 6. Whenever a macroflow γ_i is accepted, the objective value of Eq. (15) will be increased with $f(\gamma_i)$. According to Lemma 10, the objective value of the dual problem increases less than $(1+\epsilon)f(\gamma_i)$. Therefore, the overall objective value of Eq. (15) is at least $(1-\epsilon)$ times as that of Eq. (16). We thus know that the network throughput of PD-RSC is at least $(1-\epsilon)\cdot$ OPT, where OPT denotes the optimal result for the online RSC problem.

According to Eq. (19), we observe that if a link e satisfies $\alpha(e)>1$, macroflow γ_i will be rejected on this link. Thus the load on link e will not increase anymore. This means before the last update of $\alpha(e)$, we have $\alpha(e)\leq 1$. According to Eq. (20), we know in any iteration of the PD-RSC algorithm, $\alpha(e)\leq 1+B^*$. Combining with the above analysis and Lemma 11, we have

$$
\begin{aligned}
\frac{L(e,\gamma_i)}{c(e)}&\leq\log\left[(1+B^*)B^*\cdot\mathcal{N}/\epsilon+1\right]\\
&=O(\log\mathcal{N}+\log(1/\epsilon))
\end{aligned}
\tag{39}
$$

This result shows that at the end of the online algorithm, the violation level on the link capacity constraint is upper bounded by $O(\log\mathcal{N}+\log(1/\epsilon))$. Based on the similar analysis, the violation levels on the switch computing capacity constraint and the rule capacity constraint are also upper bounded by $O(\log\mathcal{N}+\log(1/\epsilon))$. Therefore, our proposed PD-RSC algorithm can achieve the competitive ratio of $[1-\epsilon, O(\log\mathcal{N}+\log(1/\epsilon)]$.

REFERENCES

[1] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] *The Openflow Switch*. Accessed: Jun. 24, 2020. [Online]. Available: http://openflowswitch.org

[3] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. SIGCOMM SIGCOMM*, 2013, pp. 15–26.

[4] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.

[5] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE INFOCOM*, Apr. 2013, pp. 2211–2219.

[6] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, Oct. 2014, pp. 563–568.

[7] H. Wang, H. Xu, L. Huang, J. Wang, and X. Yang, "Load-balancing routing in software defined networks with multiple controllers," *Comput. Netw.*, vol. 141, pp. 82–91, Aug. 2018.

[8] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Sep. 2013.

[9] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Proc. 16th Int. Conf. Adv. Commun. Technol.*, Feb. 2014, pp. 167–171.

[10] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *Proc. 22nd Telecommun. Forum Telfor (TELFOR)*, Nov. 2014, pp. 111–114.

[11] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent SDN based traffic (de)aggregation and measurement paradigm (iSTAMP)," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 934–942.

[12] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "FlowCover: Low-cost flow monitoring scheme in software defined networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1956–1961.

[13] B. Pfaff *et al.*, "Openflow switch specification v1.3.0." Open Netw. Found., Menlo Park, CA, USA, Tech. Rep. ONF TS-006, 2012.

[14] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 1734–1742.

[15] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf. IMC*, 2009, pp. 202–208.

[16] G. Zhao, H. Xu, S. Chen, L. Huang, and P. Wang, "Joint optimization of flow table and group table for default paths in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1837–1850, Aug. 2018.

[17] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 245–256, 2004.

[18] P. Phaal and M. Lavine. (2004). *Sflow version 5*. [Online]. Available: http://www.sflow.org/sflow_version_5.txt

[19] Y. Li, R. Miao, C. Kim, and M. Yu, "Flowradar: A better netflow for data centers," in *Proc. NSDI*, 2016, pp. 311–324.

[20] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, Apr. 2005.

[21] G. Cormode, "Sketch techniques for approximate query processing," in *Foundations and Trends in Databases*. NOW Publishers, 2011.

[22] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with UnivMon," in *Proc. Conf. ACM SIGCOMM Conf. SIGCOMM*, 2016, pp. 101–114, doi: 10.1145/2934872.2934906.

[23] Q. Huang *et al.*, "SketchVisor: Robust network measurement for software packet processing," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 113–126, doi: 10.1145/3098822.3098831.

[24] X. Yu, H. Xu, D. Yao, H. Wang, and L. Huang, "CountMax: A lightweight and cooperative sketch measurement for software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2774–2786, Dec. 2018.

[25] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement.*, 2013, pp. 29–42.

[26] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *Proc. Joint Int. Conf. Meas. Modeling Comput. Syst. SIGMETRICS /PERFORMANCE*, 2004, pp. 177–188.

[27] T. Koponen *et al.*, "Network virtualization in multi-tenant datacenters," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 203–216.

[28] M. Alaluna, E. Vial, N. Neves, and F. M. V. Ramos, "Secure multi-cloud network virtualization," *Comput. Netw.*, vol. 161, pp. 45–60, Oct. 2019.

[29] *Openvswitch*. Accessed: Jun. 24, 2020. [Online]. Available: http://openvswitch.org

[30] Accessed: Jun. 24, 2020. [Online]. Available: http://technet.microsoft.com/en-us/library/hh831823.aspx

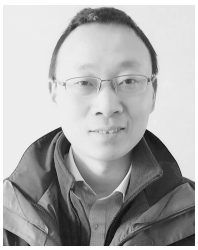
[31] *Cisco Nexus 1000V Switch*. Accessed: Jun. 24, 2020. [Online]. Available: http://www.cisco.com/c/en/us/support/switches/nexus-1000v-kvm/series.html

[32] Accessed: Jun. 24, 2020. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/28032/intel-xeon-processor-e5345-8m-cache-2-33-ghz-1333-mhz-fsb.html

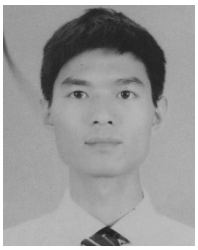
[33] I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10 Gbps FPGA-based network intrusion detection system," in *Proc. Int. Conf. Field Program. Logic Appl.* Lisbon, Portugal: Springer, Sep. 2003, pp. 880–889.

[34] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[35] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, "SilkRoad: Making stateful layer-4 load balancing fast and cheap using switching ASICs," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 15–28.
[36] Y. Takahashi *et al.*, "Separating predictable and unpredictable flows via dynamic flow mining for effective traffic engineering," *IEICE Trans. Commun.*, vol. 101, no. 2, pp. 538–547, 2018.
[37] H. Xu, Z. Yu, C. Qian, X.-Y. Li, and Z. Liu, "Minimizing flow statistics collection cost of SDN using wildcard requests," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[38] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 206–217, 2003.
[39] D. Jiang, X. Wang, and L. Guo, "An optimization method of large-scale IP traffic matrix estimation," *AEU Int. J. Electron. Commun.*, vol. 64, no. 7, pp. 685–689, Jul. 2010.
[40] A. Soule, K. Salamatian, A. Nucci, and N. Taft, "Traffic matrix tracking using Kalman filters," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 3, pp. 24–31, Dec. 2005.
[41] T. Thomas, *Ospf Network Design Solutions*. Indianapolis, IN, USA: Cisco Press Publications, 2003.
[42] G. P. Katsikas, T. Barbette, D. Kostic, R. Steinert, and G. Q. Maguire, Jr, "Metron: NFV service chains at the true speed of the underlying hardware," in *Proc. 15th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2018, pp. 171–186.
[43] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proc. 16th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1975, pp. 184–193.
[44] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, Dec. 1987.
[45] L. Guo, J. Pang, and A. Walid, "Joint placement and routing of network function chains in data centers," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 612–620.
[46] F. Guillemin and J. Boyer, "Analysis of the m/m/1 queue with processor sharing via spectral theory," *Queueing Syst.*, vol. 39, no. 4, pp. 377–397, 2001.
[47] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun. SIGCOMM*, 2008, pp. 63–74.
[48] *The Network Topology From the Monash University*. Accessed: Jun. 24, 2020. [Online]. Available: http://www.ecse.monash.edu.au/twiki/bin/view/InFocus/LargePacket-switchingNetworkTopologies
[49] I. Antoniou, V. V. Ivanov, V. V. Ivanov, and P. V. Zrelov, "On the log-normal distribution of network traffic," *Phys. D, Nonlinear Phenomena*, vol. 167, nos. 1–2, pp. 72–85, Jul. 2002.
[50] H. Xu, H. Huang, S. Chen, and G. Zhao, "Scalable software-defined networking through hybrid switching," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[51] H. Xu, H. Huang, S. Chen, G. Zhao, and L. Huang, "Achieving high scalability through hybrid switching in software-defined networking," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 618–632, Feb. 2018.
[52] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Proc. Int. Workshop Randomization Approximation Techn. Comput. Sci.* Cambridge, MA, USA: Springer, Sep. 2002 pp. 1–10.

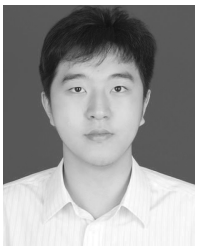
Yutong Zhai received the B.S. degree in computer science from the University of Science and Technology of China in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include software defined networking, Internet traffic measurements, and edge computing.



Hongli Xu (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China (USTC), China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, USTC. He has published more than 100 articles in famous journals and conferences, including the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Infocom, ICNP, and so on. He has also held more than 30 patents. His main research interests include software defined networks, edge computing, and the Internet of Thing. He was awarded the Outstanding Youth Science Foundation of NSFC in 2018. He has won the best paper award or the best paper candidate in several famous conferences.



Haibo Wang (Student Member, IEEE) received the B.E. degree in nuclear science and the master's degree in computer science from the University of Science and Technology of China in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in computer and information science and engineering with the University of Florida. His main research interests include Internet traffic measurements, software defined networks, and optical circuit scheduling.



Zeyu Meng received the B.S. degree from Sun Yat-sen University in 2016. He is currently pursuing the Ph.D. degree with the School of Cyberscience, University of Science and Technology of China. His main research interests include software defined networks, edge computing, and networking for AI.

He Huang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2011. He is currently a Professor with the School of Computer Science and Technology, Soochow University, China. His current research interests include traffic measurements, computer networks, and algorithmic game theory. He is a member of the ACM.