Distributed Strategy for Collaborative Traffic Measurement in a Multi-Controller SDN

Da Yao, Qianpiao Ma, Haibo Wang, Min Chen, *Graduate Student Member, IEEE*, and Hongli Xu, *Member, IEEE*

***Abstract*—Traffic measurement provides fundamental flow statistics for a wide range of network applications. In traditional networks, since switches work independently, a flow may be measured by multiple switches along its route path, which is a waste of network resource. Software defined networking (SDN) is a new paradigm that decouples control plane and data plane. The control plane is implemented in a centralized controller to manage all switches with a global view. Many works focus on designing efficient centralized strategy to coordinate the switches for traffic measurement, called collaborative traffic measurement. However, with the continuous growth of the network topology and traffic volume, the controller may be the bottleneck. To improve the capability and enhance the robustness of control plane, multiple controllers is employed to manage a subset of switches, respectively. Unfortunately, control plane with multiple distributed controllers cannot execute the centralized measurement strategy. In this paper, we propose a novel distributed iterative strategy for collaborative traffic measurement in a multi-controller SDN. We also theoretically prove that our proposed algorithm will converge to the global optimal by iterations. The extensive simulations demonstrate that the proposed strategy achieves a near-optimal performance (gap about 10%) in terms of measurement load at switches and drastically reduces the communication load by 20% at least among controllers compared with the state-of-the-art algorithms.**

***Index Terms*—Collaborative traffic measurement, multiple controllers, software defined networks.**

I. Introduction

Traffic measurement collects fundamental flow statistics in the data plane, which is the basis for the control plane to manage the network and well serve a number of applications, including load balance [1], [2], [3], flow rerouting [4], [5], fairness [6], instruction detection [7], [8], anomaly detection [9], [10], [11], traffic engineering [12], performance diagnostics [13] and policy enforcement [14].

In traditional networks, as each switch (or router) has its own data plane and control plane, it usually performs traffic measurement independently, called *independent traffic measurement*. For instance, each switch may mirror a portion p of the packet stream that passes through it (e.g., NetFlow). As another example, each switch may record all the incoming packets and perform approximate measurement using the compact data structures called *sketches* [15], [16], [17]. However, since each packet may traverse multiple switches along its route path, the packet may be recorded by multiple or all switches, leading to redundant traffic measurement. Because of the redundancy, the portion of new traffic statistics collected by a switch is actually less than p (for NetFlow), resulting in memory/computation inefficiency. For sketch-based traffic measurement, the redundancy allows the packets recorded elsewhere to be recorded in the local sketch, resulting in computation inefficiency and degrading the estimation accuracy.

To avoid measurement redundancy, we need to perform *collaborative traffic measurement*, where one switch will not repeat measuring flows that are measured by other switches along their route paths and in the meanwhile one switch must measure the flows if there no other optional switches. This kind of collaboration can hardly be supported by traditional non-SDN networks but can be realized in Software-Defined Networks (SDN). SDN decouples the control plane from switches. The control plane is centralized in a single controller with a network-wide view, which manages all the switches in the SDN. It can design an efficient *measurement strategy* to coordinate all the switches and to avoid redundancy. *Note that this paper focuses on measurement strategies rather than measurement techniques.*

There are some works [18], [19], [20] proposing efficient strategies for collaborative traffic measurement. cSamp [18] assumes that the centralized controller can gather traffic information and route information for each origin-destination (OD) pair. Specifically, the controller is input the number of flows and route path for each OD pair. By solving a linear program with the objective of the minimizing the maximum measurement load on any switch, the controller derives sampling probability of any flow at any switch along its routing path. The sampling probability will be sent to the switch for sampling process of each incoming packet in that flow. For any switch, it needs a table of size $O(n^2)$ to maintain the sampling probabilities for all flows passing through it, where n is the number of ingress/egress

Manuscript received 15 December 2022; revised 13 March 2023; accepted 24 April 2023. Date of publication 27 April 2023; date of current version 30 April 2024. This work was supported in part by the National Science Foundation of China (NSFC) under Grants 62102392 and 62132019. Recommended for acceptance by Dr. Haipeng Dai. *(Corresponding authors: Hongli Xu; Qianpiao Ma.)*

Da Yao, Min Chen, 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 Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China (e-mail: yddzf@mail.ustc.edu.cn; mchen330@mail.ustc.edu.cn; xuhongli@ustc.edu.cn).

Qianpiao Ma is with the Purple Mountain Laboratories, Research Center for Future Networks, Nanjing 211111, China (e-mail: maqianpiao@pmlabs.com.cn).

Haibo Wang is with the Department of Computer Science, University of Kentucky, Lexington, KY 40506 USA (e-mail: wanghaibo.pro@gmail.com).

Digital Object Identifier 10.1109/TNSE.2023.3271123

switches. Instead of maintaining a table, DCM [19] employs two Bloom filters, with the first one encoding the set of flows to be measured and the second one helping remove false positives from the first filter. The Bloom filters at each switch are maintained and updated by the centralized controller to adapt to traffic dynamics. The recent work proposed by Xu et al. [20] decreases the table size at each switch from $O(n^2)$ to $O(1)$ by assigning each switch a constant sampling probability which applies to all flows passing through it.

Prior works all rely on a *centralized strategy* that is executed on a single controller for collaborative traffic measurement. As the continuous growth of network topology, traffic and application requirements [21], [22], a number of literatures, including HyperFlow [23], Onix [24] and ONOS [25], advocate the control plane of multiple controllers, each exclusively managing a subset of switches in the network. *This kind of control plane achieves high reliability and scalability, and is the scenario that this paper focuses on*. Unfortunately, all the above solutions to collaborative traffic measurement cannot be applied to the multi-controller scenario directly, because the switches are partitioned into several domains and each controller manages one of these domains. One may argue to let one of the controllers be the "super" controller which aggregates the information from all controllers and executes the centralized strategy. However, this faces three challenges as follow.

- *Controller bottleneck:* Comparing the "super" controller with an imaginary single controller that governs the whole network, both controllers need the same amount of input information and need to solve the algorithm with the same complexity. It is apparent that the "super" controller will become the bottleneck—it is because a single controller cannot manage the whole network that we employ a number of controllers.
- *Control message delay:* Before executing the centralized algorithm, the "super" controller should gather all input information (traffic and route information of all flows) from peer controllers through the control link. This will compete for the bandwidth of control link with some fundamental functions including state synchronization and link discovery, leading to that the controller must either wait for control packets with the non-negligible delay or act with incomplete information.
- *Single point of failure:* If a controller fails or gets disconnected from the system, the performance of the entire network's collaborative traffic measurement will go down and applications relying on it will be affected.

To address the above challenges, we propose an effective *distributed strategy* for the collaborative traffic measurement problem. To the best of our knowledge, this is the first paper on collaborative traffic measurement in multi-controller scenarios. Each controller only needs the traffic and route information of flows that pass through the local domain and communicate with its neighbouring controller(s) by exchanging a little information. After multiple rounds of communication between neighboring controllers, the solution provably converges to the final solution. The solution includes the sampling probability of any flow at any switch, providing a fine-grained collaborative traffic measurement. We also give the dedicate design in the data plane to support the collaborative traffic measurement function in multi-controller scenarios. We compare our solution to baselines that are modified from the existing centralized strategies [18], [19], [20] and demonstrate that it incurs lowest and near-optimal measurement load to cover all flows, which makes our new solution more practical in real-world systems. We also evaluate our solution under other important metrics to demonstrate its practicability.

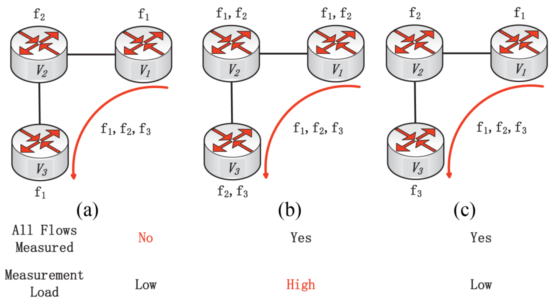


Fig. 1. Motivation Example of Collaborative Traffic Measurement. *Plot (a):* Uniform Sampling with Probability $1/3$; *Plot (b):* Uniform Sampling with Probability $2/3$; *Plot (c):* Collaborative Sampling with Probability $1/3$. Flow symbols that are placed under/over a switch icon mean that those flows are measured by the switch.

The main contributions of this paper are:

- We formulate the collaborative traffic measurement problem and propose a distributed traffic measurement probability assignment (DMPS) algorithm for the control plane, in which each controller assigns measurement fractions for switches based on its local and neighboring information rather than the whole network. We also theoretically prove that the proposed algorithm will converge to the global optimal.
- We design a dedicated packet processing mechanism in data plane to support the collaborative traffic measurement function in multi-controller scenarios.
- The extensive simulations demonstrate that the propose DMPS achieves a near-optimal performance (gap about 10%) in terms of measurement load at switches and significantly reduces the communication load by 20% at least among controllers compared with the best existing algorithms.

The rest of this paper is organized as follows. The motivating example is given to show the advantage of collaborative traffic measurement over independent traffic measurement in Section II. The system model and the problem statement is presented in Section III. The sampling probability assignment problem in the control plane is formulated and the proposed distributed algorithm is given, both in Section IV, which also includes the analysis of the proposed algorithm. Data plane packet processing that supports the collaborative traffic measurement function is described in Section V. We give the evaluation in Section VI to demonstrate its efficiency. Finally, we conclude the paper in Section VII.

II. Motivating Example

We give an example, depicted in Fig. 1, to demonstrate the advantage of the collaborative traffic measurement over

TABLE I
PROBABILITY DISTRIBUTION ALONG WITH THE NUMBER OF MEASURED FLOWS FOR INDEPENDENT TRAFFIC MEASUREMENT UNDER DIFFERENT SAMPLING PROBABILITIES p

Number of measured flows	$p = 1/3$	$p = 2/3$
0	0.03	0
1	0.26	0.01
2	0.53	0.17
3	0.18	0.82

independent traffic measurement. Assume that there are three flows f_1, f_2, f_3 all with the identical route path of switches $V_1 \rightarrow V_2 \rightarrow V_3$. In order to alleviate the impact of the traffic measurement on some important functions, e.g., packet forwarding, the traffic measurement load on the switch (evaluated by the number of measured flows) should be low.

Consider the case for independent sampling where the sampling probability is low, i.e., $1/3$. We calculate the probability that a given number of flows are measured among all switches, presented in Table I. The event that all flows are measured happens with the probability of only 18% and it is mostly likely that two flows are measured (with the probability of 53%), which is depicted in plot (a) of Fig. 1 as an example. Plot (a) shows that flow f_1 is measured twice on switches V_1 and V_3. This redundant measurement can hardly be avoided as switches perform independent sampling. To measure more flows, a trivial method is to improve the sampling probability, (e.g., $2/3$). From the third column of Table I, the event that three flows are measured among all switches happens with a high probability (e.g., 0.82). We depict plot (b) of Fig. 1 as an example, which shows that the measured flows on switch V_1 are f_1, f_2, those on switches V_2 and V_3 are f_1, f_2 and f_2, f_3. All flows f_1, f_2, f_3 are measured and the traffic measurement load of each switch is 2. However, we find that flow f_1 is measured twice by switches V_1 and V_2, and flow f_2 is measured by all three switches, resulting in measurement redundancy.

Collaborative traffic measurement in the example achieves the *ideal* results that every flow is measured and the measurement load on each switch is 1, depicted in plot (c) of Fig. 1 as an example. The sampling probability for any flow on any switch is $1/3$. By coordinating three switches V_1, V_2, V_3 along the route path of any flow, the collaborative traffic measurement guarantees that 1) any flow is sampled with the probability of $3 \times \frac{1}{3} = 1$, and 2) any flow must not be sampled multiple times. These two conditions are satisfied by some techniques for collaborative traffic measurement. In a single-controller SDN, for each switch in the route path of f, instead of assigning a value representing the sampling probability, what the controller actually assigns is a *range*. Specifically, as shown in Fig. 2, the controller splits the range $[0, 1)$ into three smaller, disjoint ranges $[0, \frac{1}{3})$, $[\frac{1}{3}, \frac{2}{3})$, $[\frac{2}{3}, 1)$. When f' packets comes at any switch, we will perform a uniform hash $H(f) \in [0, 1)$. Apparently, $H(f)$ locates in one and only one of the three ranges and f is sampled by that switch. For the case where there are multiple controllers in the control plane, the above techniques are not applicable. The reason is that a flow passes through multiple switches that may

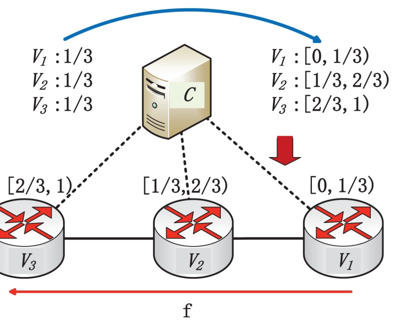


Fig. 2. Motivation Example Comparison about Sampling Probability Assignment.

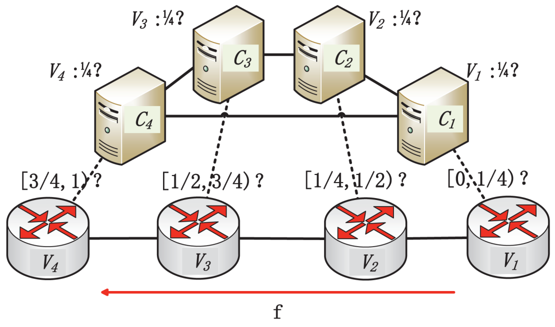


Fig. 3. Motivation Example Comparison about Sampling Probability Assignment with multiple controllers.

belong to different domains while the controller only manages local switches, depicted in Fig. 3. There are four controllers in the control plane and four switches are managed by the corresponding controller. The *ideal* sampling probability for flow f on all switches is $1/4$. However, since each controller (e.g., C_1) only has the traffic information on the corresponding switch (e.g., V_1), it is impossible to give the *ideal* sampling probability for flow f. Therefore, the range [0, 1) also cannot be split into smaller, disjoint ranges appropriately. As this paper focuses on a multi-controller scenario, the challenge of collaborative traffic measurement that the sampling probability calculation in the control plane and the sampling probability distribution in the data plane will be addressed in Section IV and Section V.

III. SYSTEM MODEL AND PROBLEM STATEMENT

This section first introduces the system model and then gives the problem statement. Some important notations are listed in Table II.

A. System Model

A classical software-defined network logically consists of a data plane and a control plane. The data plane consists of a set of switches, denoted as $V = \{v_1, v_2, \ldots, v_n\}$, $n = |V|$, which is responsible for packet forwarding and other data-plane functions, e.g., traffic measurement. The control plane is responsible for managing the entire network, including route selection/update and making rules for collaborative traffic measurement. Since we consider the multi-controller scenario,

TABLE II
NOTATION SUMMARY

Notation	Definition
V	Set of switches $\{v_1, v_2, ..., v_n\}$
C	Set of controllers $\{c_1, c_2, ..., c_m\}$
P	Set of routing paths $\{p_1, p_2, ..., p_s\}$
ϕ_j	Number of distinct flows on the route path p_j
l_i	Measurement load of switch v_i
$\widehat{C}_k$	Set of controllers including controller c_k and its neighbors
V_k	Set of switches managed by controller c_k
$\widehat{V}_k$	Set of switches managed by controllers in $\widehat{C}_k$
I	Fairness Index
$w_{i,j}$	Sampling probability of flows traversing route path p_j on switch v_i
$\widehat{w}_{i,j}^k$	Estimation of sampling probability of flows traversing route path p_j on switch v_i by controller c_k

the set of m controllers in denoted as $C = \{c_1, c_2, \ldots, c_m\}, m = |C|$. The set of switches managed by each controller c_k is called a *domain*, which is denoted as $V_k, 1 \le k \le m$, with $n_k = |V_k|$. Due to high synchronization cost and possible state inconsistence issue, we assume each switch is managed by one and only one controller, i.e., $V_i \bigcap V_j = \emptyset, \forall i, j \in [1, m], i \neq j$. Each controller can communicate directly with its neighbour controller(s) with east-west interfaces [26], [27], [28]. Moreover, we denote the set of controller c_k and its neighbour(s) as $\widehat{C}_k$, and the set of switches managed by the controllers in $\widehat{C}_k$ is denoted as $\widehat{V}_k$.

The traffic of the network is modeled as network flows, which are identified by one or multiple fields carried in the packet header, depending on the application needs. A typical flow identifier is a 5-tuple information containing source address, source port, protocol, destination address, and destination port. The set of the route paths for flows in the network is denoted as $P = \{p_1, p_2, \ldots p_s\}$ with $s = |P|$. Each route path of flows in the network consists of a sequence of switches in the network, which is denoted as p_j. We assume that each controller has a knowledge of the global route path of the flows that appears in the local domain. This assumption can be supported by exchanging a little information from peer controllers. Sometimes, if the routing policy is simple or straight forward, the controller can derive the route path of the flow in the whole network itself. From the route paths of each flow, we know the number of flows on route path p_j, denoted as ϕ_j. The variable $w_{i,j} \in [0, 1]$ denotes the sampling probability of flows on switch v_i traveling through its route path p_j. Let $[p_j]$ be the set of switches on path p_j. The measurement load l_i of switch v_i is defined as the number of measured flows on it.

B. Problem Statement

The problem studied in this paper is called *flow distribution* problem. Specifically, for any flow appearing in the local domain of each controller $c_k \in C$, i.e., $p_j \bigcap V_k \neq \varnothing$, the controller needs to decide whether the flow is measured by one of the switches in its local domain (the flow may traverse multiple domains and thus may not be necessarily measured locally). If the flow is measured in its local domain, the controller should decide on which switch the flow is measured. The constraint is that any flow must be measured by one (for the purpose of improving measurement performance) and only one switch (for the purpose of reducing the total overhead) in the whole multi-controller SDN. The objective is to balance the measurement load among all switches.

The flow distribution problem will be solved by the measurement strategy design in the control plane and the dedicate packet processing in the data plane, which are given in Section IV and Section V.

IV. Problem Formulation for Sampling Probability Assignment on Control Plane

The sampling probability assignment problem studies how the controller assigns the appropriate traffic measurement probabilities for each flow to the switches along its route path, in order to make the traffic measurement load balanced among all switches in the entire network. That is, each flow is measured by one and only one switch such that the measurement load among switches is balanced. The measurement load of switch v_i is defined as the number of measured flows on it, denoted as l_i. The traffic measurement load on switch v_i can be calculated by

$$l_i = \sum_{p_j \in P} w_{i,j} \phi_j \tag{1}$$

Based on the traffic measurement load on switch v_i, the fairness index I [29], indicating the measurement load balance among all switches, can be expressed as

$$I = \frac{(\sum_{v_i \in V} l_i)^2}{n \cdot \sum_{v_i \in V} l_i^2} \tag{2}$$

The measurement load balance is optimally achieved when $l_i = l_{i'}$, $\forall v_i \neq v_{i'}$. In this case, the fairness index $I = 1$. The load balanced is degraded if I decreases. The worst case is that all flows are measured at one switch and the rest of switches are idle, i.e., $l_{i'} = 0, \forall v_{i'} \neq v_i$ and $l_i \neq 0$, resulting in $I = \frac{1}{n}$. To balance the measurement load among all switches, we expect to maximize the value of fairness index I. Therefore, we formulate our distributed traffic measurement probability assignment (DMPS) problem as follows:

$$(\mathbf{P1}): \max\ I \tag{3a}$$

$$\text{s.t} \sum_{v_i \in p_j} w_{i,j} = 1, \qquad \forall p_j \in P \tag{3b}$$

$$0 \le w_{i,j} \le 1, \qquad \forall v_i \in V, p_j \in P \tag{3c}$$

The first (3b) ensures that each flow in the network will be measured by one switch along the route path p_j. The objective of DMPS is to balance the traffic measurement loads among all switches, i.e., maximize the fairness index of all switches in the entire network, $\max\ I$.

The main challenge to derive the optimal solution of $\mathbf{P1}$ is the incompleteness of traffic information for a controller in a multi-controller scenario. It is difficult for a single controller to assign the appropriate traffic measurement probability for flows on switch v_i along its route path p_j, because it only has the traffic information on the switches in its local management area. In this paper, we propose a distributed traffic measurement probability assignment (DMPS) algorithm in multi-controller scenario to balance the traffic measurement loads among all switches.

A. Algorithm Description

The algorithm runs in a distributed manner, and without loss of generality we consider an arbitrary controller c_k. We consider a time series of $0, 1, \ldots, t \ldots$. Let $w_{i,j}^{(t)}$ denote the sampling probability of flows on switch v_i along path p_j at iteration t. Initially, $w_{i,j}^{(0)}$ is set as $\frac{1}{|p_j|}$, where $|p_j|$ is the number of switches in routing path p_j (this information is known for any controller as we have assumed that control plane knows routing path of each flow). The algorithm mainly consists of the following four steps at iteration t.

Step 1: Controller c_k collects $w_{i,j}^{(t)}$ for all switches $v_i \in \hat{V}_k$ and all flows that appears in $\hat{C}_k$ (Lines 3-4). In practice, this can be done through controllers' communication, e.g., the west-east protocol [26], [27], [28].

Step 2: Based on the collected sampling probability $w_{i,j}^{(t)}$ for all switches in $\hat{V}_k$, the controller c_k will solve problem **P2** to obtain the $w_{i,j}^{k,(t+1)}$, which is called *selfish measurement fraction* for c_k and is a temporary constant (Line 5). The problem **P2**, denoted as the neighbor optimization stage, is presented in Section IV-B.

Step 3: c_k will send $w_{i,j}^{k,(t+1)}$ to its neighbor controller(s) that v_i belongs to (Lines 6–7).

Step 4: after receiving $w_{i,j}^{k',(t+1)}$ from each neighbor controller $c_{k'}$, c_k will compute $w_{i,j}^{(t+1)}$ (Lines 8–11).

After all controllers receive the new sampling probabilities $\widehat{w}_{i,j}^{k,(t+1)}$, the last strategy for sampling probability update, expressed by (5), will be applied to obtain the sampling probabilities $w_{i,j}^{(t+1)}$ (Line 9–11). The above four steps repeat until $t \leq T$, where T is an integer constant. The setting of T should make the objective of P1 converges to close-optimum one. The value of T is usually pre-given and we will experimentally show that T is usually less than 5. The DMPS algorithm is formally described in Algorithm 1.

B. Neighbor Optimization Stage

Since one controller can only manage the switches in its management area in the multi-controller scenario, the traffic measurement load balance cannot be achieved based on its local traffic information. Therefore, the idea of neighbor optimization is proposed to simply exchange the traffic measurement loads among all switches managed by controllers in $\widehat{C}_k$, which finally makes the traffic measurement loads balanced among all switches in the entire network. The convergence proof of this method will be presented in Section IV-C. The detailed description about the neighbour optimization stage is presented below.

Algorithm 1: DMPS Algorithm at Controller c_k.

```
Input: Controller Set $\widehat{C}_k$, Route Path Set $P$
Output: Final Traffic Sampling Probabilities $w_{i,j}^{(t+1)}$
 1: $t = 0$
 2: while $t < T$ do
 3:   for each switch $v_i \in V_{k'}, p_j \in P$ do
 4:     Obtain traffic sampling probability $w_{i,j}^{(t)}$
 5:   end for
 6:   Solve P2 to obtain $\widehat{w}_{i,j}^{k,(t+1)}, \forall v_i \in \widehat{V}_k, p_j \in P$
 7:   for each controller $c_{k'} \in \widehat{C}_k$ do
 8:     Send $\widehat{w}_{i,j}^{k,(t+1)}, \forall v_i \in \widehat{V}_k, p_j \in P$
 9:   end for
10:   Receive $\widehat{w}_{i,j}^{k',(t+1)}, \forall v_i \in \widehat{V}_k, p_j \in P$ from $c_{k'} \in \widehat{C}_k$
11:   for each switch $v_i \in V_k, p_j \in P$ do
12:     Calculate $w_{i,j}^{(t+1)}$ by (5)
13:   end for
14:   $t = t + 1$
15: end while
```

Based on (2), the expression $\frac{(\sum_{v_i \in V} l_i)^2}{n}$ in the fairness index I is a constant, because $\sum_{v_i \in V} l_i$ is the total number of flows which is fixed. Therefore, we can obtain the maximum fairness index I as long as the value of the expression $\sum_{v_i \in V} {l_i}^2$ is minimized. Therefore, during the neighbour optimization stage, we can set the objective function as $\min \sum_{v_i \in \widehat{V}_k} {l_i}^2$. That is, the problem **P2** is formulated as follow:

$$(\mathbf{P2}):\ \min \sum_{v_i \in \widehat{V}_k} (\hat{l}_i^{(t+1)})^2 \tag{4a}$$

$$\text{s.t.} \sum_{v_i \in \widehat{V}_k} \widehat{w}_{i,j}^{k,(t+1)} = \sum_{v_i \in \widehat{V}_k} w_{i,j}^{(t)}, \forall p_j \in P \tag{4b}$$

$$0 \leq \widehat{w}_{i,j}^{k,(t+1)} \leq 1, \forall v_i \in \widehat{V}_k, p_j \in P \tag{4c}$$

The first (4b) ensures that for each flow the sum of the the sampling probability on switches managed by controllers in $\widehat{C}_k$ along the route path remains unchanged. The objective of problem **P2** is to balance the traffic measurement loads among all switches managed by controllers in $\widehat{C}_k$.

Note that, controller c_k obtains the traffic measurement probabilities $w_{i,j}^{(t)}$ on switches managed by controllers in $\widehat{C}_k$ at the iteration t. However, controller c_k can only know the traffic measurement probabilities on switches managed by controllers in $\widehat{C}_k$ at each iteration. Therefore, it is assumed that all the traffic measurement probabilities on switches should be updated at the same iteration. Moreover, $\hat{w}_{i,j}^{k,(t+1)}$ is the traffic measurement probabilities on switches managed by controllers in $\widehat{C}_k$ at iteration $t+1$, which is computed by controller c_k. Obviously, each controller tends to reduce the traffic measurement loads on switches managed by itself, and we called $\hat{w}_{i,j}^{k,(t+1)}$ as a selfish traffic measurement probabilities. Therefore, there is no guarantee that the traffic measurement loads among all switches will be balanced, when all controllers execute the selfish strategy for switches managed by themselves. In order to achieve the goal of traffic measurement load balanced among all switches, the step size β is applied during the adjustment of the traffic measurement probabilities on switches managed by controllers in $\widehat{C}_k$. That is, the traffic measurement probabilities $w_{i,j}^{(t+1)}$ on switch v_i at iteration $t+1$ is expressed as

$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \beta \cdot \sum_{c_k \in \widehat{C}_i} \left(\widehat{w}_{i,j}^{k,(t+1)} - w_{i,j}^{(t)} \right) \tag{5}$$

C. Convergence Analysis

In this section, we prove that the proposed algorithm converges to the approximate optimal solution of problem **P1**. For convenience, we convert **P1** to minimize the inverse of the fairness index $I(\boldsymbol{w})$, which is denoted as $J(\boldsymbol{w})$. Therefore, we can obtain

$$J(\boldsymbol{w}) = \frac{1}{I(\boldsymbol{w})} = \frac{n\sum_{v_i \in V}(l_i)^2}{(\sum_{v_i \in V} l_i)^2}, \tag{6}$$

where $\boldsymbol{w} = \{w_{i,j} \| \forall v_i \in V, p_j \in P\}$ is the traffic measurement probability assignment strategy in the network. Besides, we denote the traffic measurement probability assignment strategy at iteration t as $\boldsymbol{w}^{(t)}$. Therefore,

$$J(\boldsymbol{w}^{(t)}) = \frac{n\sum_{v_i \in V}(l_i^{(t)})^2}{(\sum_{v_i \in V} l_i^{(t)})^2} = \frac{n}{\Phi^2}\sum_{v_i \in V}(l_i^{(t)})^2, \tag{7}$$

where $\Phi = \sum_{v_i \in V} l_i^{(t)}$ and $l_i^{(t)} = \sum_{\gamma_j \in \Gamma} w_{i,j}^{(t)} \phi_j$.

According to the definition of $\boldsymbol{w}^{(t)}$, we can obtain the new traffic measurement probabilities assignment strategy $\boldsymbol{w}^{(t+1)}$ at iteration $t+1$ after the execution of the DMPS algorithm. Therefore, the transformation model of the traffic measurement probability assignment strategy $\boldsymbol{w}$ can be expressed as

$$\boldsymbol{w}^{(t+1)} = \mathcal{T}(\boldsymbol{w}^{(t)}) \tag{8}$$

Based on the above analysis, if $\boldsymbol{w}^* = \mathcal{T}(\boldsymbol{w}^*)$, the traffic measurement probabilities assignment strategy $\boldsymbol{w}^*$ can be treated as a fixed point of iterative convergence of (8).

Lemma 1: If $\boldsymbol{w}^{(t)} \neq \boldsymbol{w}^*$, $\boldsymbol{w}^{(t+1)}$ gives a descent direction at $\boldsymbol{w}^{(t)}$ for $J(\boldsymbol{w}^{(t)})$, i.e.,

$$\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle < 0,$$

where symbol ∇ denotes the gradient operator, and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Proof: According to (7), we compute the partial derivative of $J(\boldsymbol{w}^{(t)})$ with respect to $w_{i,j}^{(t)}$ as

$$\frac{\partial J(\boldsymbol{w}^{(t)})}{\partial w_{i,j}^{(t)}} = \frac{2n}{\Phi^2} l_i^{(t)} \phi_j \tag{9}$$

Therefore, we can obtain

$$\begin{aligned}
&\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle \\
&= \frac{2n}{\Phi^2} \sum_{w_{i,j}^{(t)} \in \boldsymbol{w}^{(t)}} l_i^{(t)} \phi_j \left(w_{i,j}^{(t+1)} - w_{i,j}^{(t)} \right) \\
&= \frac{2n}{\Phi^2} \sum_{v_i \in V} l_i^{(t)} \sum_{\gamma_j \in \Gamma} \phi_j \left(w_{i,j}^{(t+1)} - w_{i,j}^{(t)} \right) \\
&= \frac{2n\beta}{\Phi^2} \sum_{v_i \in V} l_i^{(t)} \sum_{\gamma_j \in \Gamma} \phi_j \sum_{c_k \in \widehat{C}_i} \left(\widehat{w}_{i,j}^{k,(t+1)} - w_{i,j}^{(t)} \right) \\
&= \frac{2n\beta}{\Phi^2} \sum_{v_i \in V} \sum_{c_k \in \widehat{C}_i} l_i^{(t)} \sum_{p_j \in P} \phi_j \left(\widehat{w}_{i,j}^{k,(t+1)} - w_{i,j}^{(t)} \right) \\
&= \frac{2n\beta}{\Phi^2} \sum_{v_i \in V} \sum_{c_k \in \widehat{C}_i} l_i^{(t)} \left(\hat{l}_i^{k,(t+1)} - l_i^{(t)} \right) \\
&= \frac{2n\beta}{\Phi^2} \sum_{c_k \in C} \sum_{v_i \in \widehat{V}_k} l_i^{(t)} \left(\hat{l}_i^{k,(t+1)} - l_i^{(t)} \right) \\
&= \frac{2n\beta}{\Phi^2} \sum_{c_k \in C} \left(\sum_{v_i \in \widehat{V}_k} l_i^{(t)} \cdot \hat{l}_i^{k,(t+1)} - \sum_{v_i \in \widehat{V}_k} (l_i^{(t)})^2 \right)
\end{aligned} \tag{10}$$

According to (4a), we can obtain the inequality as follow

$$\sum_{v_i \in \widehat{V}_k} \left(\hat{l}_i^{(t+1)} \right)^2 \leq \sum_{v_i \in \widehat{V}_k} \left(l_i^{(t)} \right)^2 \tag{11}$$

Therefore, we can deduce that

$$\begin{aligned}
\sum_{v_i \in \widehat{V}_k} l_i^{(t)} \cdot \hat{l}_i^{k,(t+1)} &\leq \frac{1}{2} \left[\sum_{v_i \in \widehat{V}_k} (l_i^{(t)})^2 + \sum_{v_i \in \widehat{V}_k} \left(\hat{l}_i^{k,(t+1)} \right)^2 \right] \\
&\leq \sum_{v_i \in \widehat{V}_k} \left(l_i^{(t)} \right)^2
\end{aligned} \tag{12}$$

On one hand, by (9), we can obtain that the function $J(\boldsymbol{w}^{(t)})$ is strictly convex. That is, as long as $\boldsymbol{w}^{(t)} \neq \boldsymbol{w}^*$, we can obtain $\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle < 0$.

On the other hand, combining (10) and (12), if $\sum_{v_i \in \widehat{V}_k} (\hat{l}_i^{(t+1)})^2 = \sum_{v_i \in \widehat{V}_k} (l_i^{(t)})^2$ for all controllers c_k, the gradient function $\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle = 0$. That is, when the gradient function $\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle = 0$, we can obtain that $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$, and $\boldsymbol{w}^{(t)} = \boldsymbol{w}^*$.

According to the above conclusion, we can conclude that the gradient function $\langle \nabla J(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} \rangle < 0$. □

Lemma 2: If $\boldsymbol{w}^{(t)} \neq \boldsymbol{w}^*$, there exists $\beta \in (0,1]$ such that $J(\boldsymbol{w}^{(t+1)}) < J(\boldsymbol{w}^{(t)})$.

Proof: According to (5), we can regard $\boldsymbol{w}^{(t+1)}$ as a variable vector that varies with the value of parameter β, denoted as $\boldsymbol{w}^{(t+1)}(\beta)$. Therefore, $\boldsymbol{w}^{(t)}$ becomes a constant vector with the parameter $\beta = 0$, which means the traffic measurement probabilities assignment strategy is same as that at any iteration t, i.e., $\boldsymbol{w}^{(t+1)}(0) = \boldsymbol{w}^{(t)}$.

By the definition of $J(\boldsymbol{w}^{(t)})$ in (7), $J(\boldsymbol{w}^{(t+1)})$ is differentiable for each component of $\boldsymbol{w}^{(t+1)}$. Since (5) is a linear transformation, the components of $\boldsymbol{w}^{(t+1)}$ can be differentiable on the parameter β. Thus, we can deduce that $J(\boldsymbol{w}^{(t+1)})$ is differentiable on the parameter β, too.

To simplify the expression, we define $G(\beta) = J(\boldsymbol{w}^{(t+1)}(\beta))$ as a function of the parameter β. Therefore, we can obtain that $G(0) = J(\boldsymbol{w}^{(t+1)}(0)) = J(\boldsymbol{w}^{(t)})$.

Assume that $\forall\beta \in (0, 1]$, the expression $G(\beta) \geq G(0)$ always exists, i.e., $J(\boldsymbol{w}^{(t+1)}) \geq J(\boldsymbol{w}^{(t)})$.

On one hand, by chain rule, we can obtain that

$$\begin{aligned} G'_\beta &= \sum_{w_{i,j}\in \boldsymbol{w}^{(t+1)}} \frac{\partial J}{\partial w_{i,j}^{(t+1)}} \cdot \frac{\partial w_{i,j}^{(t+1)}}{\partial \beta} \\ &= \langle \nabla J(\boldsymbol{w}^{(t+1)}), (\boldsymbol{w}^{(t+1)})'_\beta \rangle \end{aligned}$$

By Lemma 1, for $\forall\beta \to 0^+$, we can obtain that

$$\begin{aligned} G'(0^+) &= G'_\beta|_{\beta\to 0^+} \\ &= \lim_{\beta\to 0^+} \left\langle \nabla J(\widetilde{\boldsymbol{w}}^{(t+1)}), \frac{\widetilde{\boldsymbol{w}}^{(t+1)}(\beta) - \widetilde{\boldsymbol{w}}^{(t+1)}(0)}{\beta} \right\rangle \\ &= \lim_{\beta\to 0^+} \frac{\left\langle \nabla J(\boldsymbol{w}^{(t)}), \widetilde{\boldsymbol{w}}^{(t+1)} - \boldsymbol{w}^{(t)} \right\rangle}{\beta} \\ &< 0 \end{aligned} \tag{13}$$

On the other hand, we can obtain that

$$G'(0^+) = \lim_{\beta\to 0^+} \frac{G(\beta) - G(0)}{\beta - 0} \geq 0 \tag{14}$$

Based on the above two cases, we can find that the (13) and (14) are contradictory. As a result, the assumption that $G(\beta) \geq G(0)$ always exists with $\forall\beta \in (0, 1]$ does not hold. That is, there exists $\beta \in (0, 1]$ such that $J(\boldsymbol{w}^{(t+1)}) < J(\boldsymbol{w}^{(t)})$ □.

Theorem 1: Suppose that the parameter $\beta \in (0, 1]$ makes $J(\boldsymbol{w}^{(t+1)}) < J(\boldsymbol{w}^{(t)})$. Then, $\boldsymbol{w}^{(t)}$ can converge to $\boldsymbol{w}^*$, i.e., $\lim_{t\to+\infty} \boldsymbol{w}^{(t)} = \boldsymbol{w}^*$. In addition, $\boldsymbol{w}^*$ is the optimal solution of problem **P1**.

Proof: According to Lemma 2, $J(\boldsymbol{w}^{(t)})$ is a monotonically decreasing sequence with respect to the iteration t and $J(\boldsymbol{w}^{(t)}) > 0$. Assume that $\lim_{t\to+\infty} \boldsymbol{w}^{(t)} = \boldsymbol{w}'$ and $\boldsymbol{w}' \neq \boldsymbol{w}^*$.

According to Lemma 2, if $\boldsymbol{w}' \neq \boldsymbol{w}^*$, there exists $\beta \in (0, 1]$ such that $J(\boldsymbol{w}')$ decreases again. Therefore, we can obtain that $\lim_{t\to+\infty} \boldsymbol{w}^{(t)} = \boldsymbol{w}^*$.

In addition, assume that there exists $\boldsymbol{w}''$ such that $\mathcal{T}(\boldsymbol{w}'') \neq \boldsymbol{w}''$, and $\boldsymbol{w}''$ is the vector with the minimum value of $J(\boldsymbol{w})$.

According to Lemmas 1 and 2, $\mathcal{T}(\boldsymbol{w}'')$ gives a descent direction at $\boldsymbol{w}''$, and there exists the parameter $\beta \in (0, 1]$ such that $J(\mathcal{T}(\boldsymbol{w}'')) < J(\boldsymbol{w}'')$, which contradicts our assumption. Therefore, $\boldsymbol{w}^*$ is the optimal solution of problem **P1**. □

V. Packet Processing on Data Plane

In the multi-controller SDNs, for flows appearing in the local domain of controller c_k, the controller will assign a sampling probability $w_{i,j}$ to switch v_i along its route path p_j. How to obtain $w_{i,j}$ in the control plane has been given in Section IV. However, since the controller can only manage the switches in its local domain, there is no guarantee that each flow can be measured once on a switch along its route path. To realize flow distribution in the local domain, there need some extra lightweight operations in the data plane of switches to adapt the sampling probability $w_{i,j}$ for flows on each switch to traditional packet processing. In other words, two constraints should be satisfied:

- Flows traversing through route path p_j should be measured one and only once by any switch v_i along the route path, i.e., $\sum_{v_i\in p_j} w_{i,j} = 1$;
- The operation of sampling probability update for flows traversing through route path p_j on switch v_i should be lightweight.

We maintain a state bit in each packet header, denoted as $Flag = 0$, which indicates whether this packet has been measured ($Flag = 1$) or not ($Flag = 0$). Besides, we maintain another field in each packet header of flows, denoted as HW_j which represents the cumulative sampling probabilities from the ingress switches to the current switches (included) along the route path p_j. Initially, $HW_j = 0$ and $Flag = 0$. Note that every switch uses the hash value $\omega \in [0, 1)$ obtained by the same uniform hash function $H(\cdot)$ that takes the flow identifier as the arguments, guaranteeing consistently the same hash value at any switch as long as the the packets belong to the same flow. Our design ensures that a flow will be measured on the switch where both $Flag = 0$ and $\omega \geq HW_j$, if the value of ω is less than the value of updated HW_j. Then, Flag is updated to 1. In a word, upon a packet of flow arriving at switch v_i in its route path p_j, there are two cases:

- If $Flag = 0$, it means the flow has not been measured and we know $\omega \geq HW_j$. Then, we update HW_j to $HW_j + w_{i,j}$. There are two sub-cases.
  - If $\omega < HW_j$, we will measure the packet and update $Flag$ as 1.
  - Otherwise, the packet will be forwarded directly.
- If $Flag = 1$, it means the packet has been measured and there is no need to measure this packet. Therefore, the packet will be forwarded directly.

The design for the real-time packet processing for flows on switch v_i is illustrated in Fig. 4. Note that it will not significantly increase the overhead by adding some state information to the packet header, which has been widely used in various applications, such as middle box routing [30] and segment routing [31].

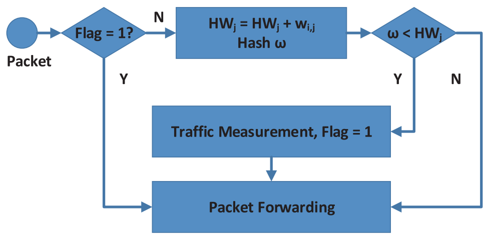


Fig. 4. Illustration of Real-time Packet Processing for Flow Distribution.

VI. Evaluation

In this section, we first introduce the benchmarks and metrics for our simulations in Section VI-A. Then we describe the simulation settings in Section VI-B. Finally, we evaluate our proposed algorithms through extensive simulations in Section VI-C.

A. Benchmarks and Performance Metrics

Since DMPS is the first studying the collaborative traffic measurement in a multi-controller SDN, there is no prior work for comparison. The most related works, i.e., cSamp [18] and NSPA [20], are centralized algorithms that are designed for collaborative traffic measurement in a single-controller SDN. We apply them to the multi-controller scenario by assuming that they are executed at one of the controllers with global information. This assumption can hardly be true in practical large-scale SDNs as one controller is limited in computing/memory resources. However, it makes us know the optimal performance in some metrics, from which we know the gap between the optimum and DMPS. We adopt the following performance metrics.

- *The maximum measurement load among all switches.* The measurement load on a switch is defined as the number of flows measured by the switch. The lower the maximum traffic measurement load among all switches is, the more balanced the measurement loads among all switches are.
- *The fairness index I.* After measurement probabilities of each flow along its route path are obtained, the number of flows measured by each switch can be determined. We calculate the fairness index, i.e., $I = \frac{(\sum_{v_i \in V} l_i)^2}{n \cdot \sum_{v_i \in V} l_i^2}$, which indicates what extent the measurement loads among all switches are balanced.
- *The maximum communication load among all controllers.* When a centralized algorithm (e.g., cSamp, NSPA) is applied in a multi-controller scenario, the "super" controller should 1) be input the information of all flows from other controllers. For each flow, the communication load is the size of the flow identifier, which is 20 Bytes for each 5-tuple flow. By multiplying the number of flows at any domain, we can obtain the input communication load at the controller; 2) Output the sampling probability of each flow at each switch to the peer controllers, the communication message for each flow at each switch includes flow identifier, switch ID, sampling probability, result in the communication load of $20 + 4 + 4 = 28$ bytes for each message exchange. By multiplying the number of messages, we can obtain the output communication load at any switch. Adding up the input and output communication load, we can obtain the total communication load at any switch. For the DMPS algorithm, controllers exchange the sampling probability for each flow at each switch along the route path. Therefore, the message format is: flow identifier, switch ID, sampling probability, result in the communication load of $20+4+4=28$ bytes for each message exchange. By multiplying the number of messages, we can obtain the communication load at any switch.
- *The average processing overhead per-packet in traffic measurement probability distribution.* Different solutions may perform different operations, e.g., hash operation, memory access, packet-header reading/writing, etc., for flow distribution. Note that memory access refers to operations (e.g., lookup and update) on a memory list, e.g., a table of hash ranges and a bloom filter. It is trivial to consider all operations together, for different operations leading to various processing overheads. By testing on our platform with CPU 3.7 GHz and OVS version 2.5.3 [32], we find that average processing overheads (e.g., about 10-30 CPU cycles) of hash operation, and memory access are usually more than that (e.g., about 1-5 CPU cycles) of packet-head reading/writing. Thus, we analyze the average number of hash operations and memory accesses per-packet in flow traffic measurement load distribution.
- *The running time* is defined as the time required to solve the algorithm on controller plane in each benchmark.

B. Simulation Settings

In the simulations, as running examples, we select two typical and practical topologies for data center networks. The first one for data center networks is the Fat-tree topology [33], which has been widely used in many data center networks. The Fat-tree topology has in total 320 switches (including 128 edge switches, 128 aggregation switches, and 64 core switches) and 1024 terminals. The second topology VL2 [34] is another data center network topology. The topology VL2 contains 240 switches, which consists of 20 core switches, 20 aggregation switches and 200 edge switches. Each edge switch sets up connections to randomly selected peer edge switches. Considering a multi-controller scenario, we determine the number of controllers as 8 and 16 for our simulations, and all switches are evenly assigned to the controllers of the control plane. Besides, we adopt ECMP for flow routing in both topologies. One commodity server, which is equipped with 1 Intel Xeon 2.10 GHz CPU (each with 22 physical cores), use 32 K/1024 K/20976 K L1-3 caches. The server runs a Linux 5.4.0-71-generic distribution. We execute each simulation 100 times on the server, and take the average of the numerical results.

C. Experimental Results

The simulation is categorized into two groups. The first group is to evaluate the impact of simulation setting on the performance of DMPS in Section VI-C1. The second group is to compare the performance of DMPS with benchmarks in Section VI-C2, i.e., cSamp, NSPA.

1) Setting of Parameter β: Considering the parameter β in our algorithm (DMPS) defined by the prior knowledge, we should choose an appropriate parameter β (e.g., 0.3) through the simulations.

We first observe the impact of parameter β on the number of iterations, when there are 200 K flows in the network. The value of parameter β varies in the range of (0.1, 1), and the step length is set as 0.05. The results in Fig. 5 show that the number of iterations decreases when parameter β increases to 0.3. When the value of parameter β is more than 0.3, we find the algorithm does not converge. We will provide the details in each round of iteration in Figs. 7–10.

Then, we also show maximum measurement load among all switches when converging of DMPS under different values of

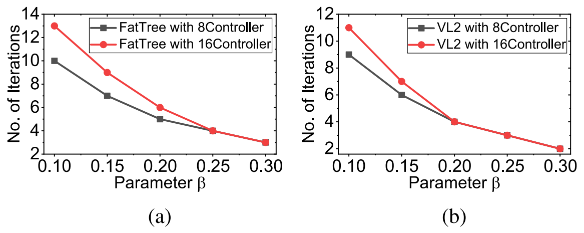


Fig. 5. Maximum number of iterations for DMPS with different values of β with respect to the number of iterations when there are 200 K flows. *Plot (a):* Topology Fat Tree; *Plot (b):* Topology VL2.

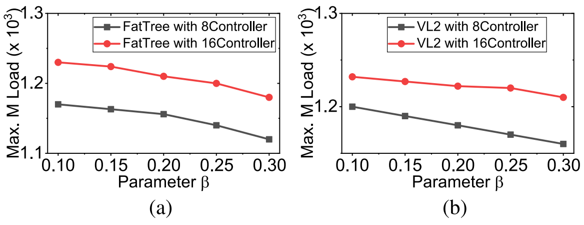


Fig. 6. Maximum measurement load for DMPS with different values of β with respect to the number of iterations when there are 200 K flows. *Plot (a):* Topology Fat Tree; *Plot (b):* Topology VL2.

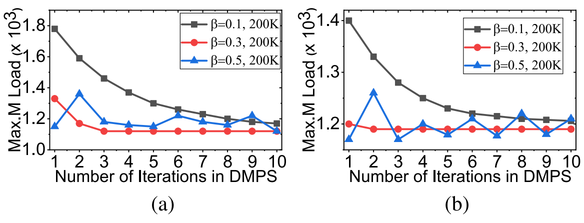


Fig. 7. Maximum measurement load of NSPA with different values of β with respect to the number of iterations when the number of flows are 200 K. *Plot (a):* Topology Fat Tree with 8 Controllers; *Plot (b):* Topology VL2 with 8 Controllers.

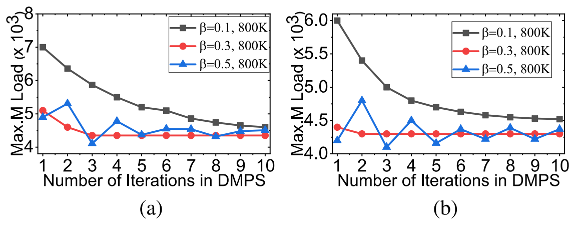


Fig. 8. Maximum measurement load of NSPA with different values of β with respect to the number of iterations when the number of flows are 800 K. *Plot (a):* Topology Fat Tree with 8 Controllers; *Plot (b):* Topology VL2 with 8 Controllers.

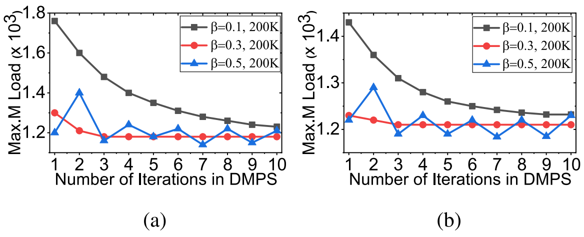


Fig. 9. Maximum measurement load of NSPA with different values of β with respect to the number of iterations when the number of flows are 200 K. *Plot (a):* Topology Fat Tree with 16 Controllers; *Plot (b):* Topology VL2 with 16 Controllers.

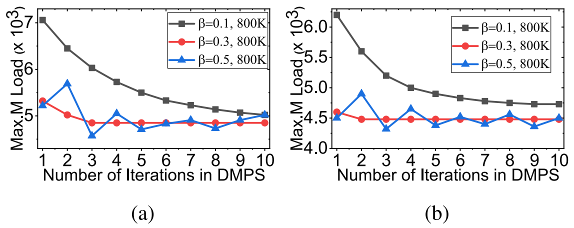


Fig. 10. Maximum measurement load of NSPA with different values of β with respect to the number of iterations when the number of flows are 800 K. *Plot (a):* Topology Fat Tree with 16 Controllers; *Plot (b):* Topology VL2 with 16 Controllers.

parameter β in DMPS. The results in Fig. 6 show that small value of parameter β will result in large maximum measurement load. The above results keep consistent when the number of flows changes. Therefore, the optimal setting about the value of parameter β is 0.3. Besides, since parameter $\beta > 0.3$ leads to the non-convergence of DMPS algorithm, the maximum measurement load among all switches is not plotted in the figure.

Last, we will take the value (0.1, 0.3, 0.5) of parameter β as representatives to evaluate the detailed performance of DMPS during execution under different topologies and different number of flows in Figs. 7–10. All figures show that the setting about the value of parameter β as 0.3 can achieve the lowest maximum measurement load with the fastest convergence speed, regardless of the network topology, compared to the setting about the value of parameter β as 0.1. Moreover, we find that the setting about the value of parameter β as 0.5 cannot obtain a stable solution because of the non-convergence of DMPS algorithm with the parameter setting. The reason is that large value of parameter β will make the controller be over-fed the sampling probabilities from neighbor controllers, i.e., the measurement load from neighbour controllers. Specifically, the plot (a) of Fig. 7 describes the maximum traffic measurement load among all switches under the topology Fat Tree with 8 controllers when there are 200 K flows in the network. When the value of parameter β is set as 0.1, the minimum value of the maximum traffic measurement load is about 1170 after 10 iterations of our DMPS algorithm. However, only 3 iterations is required to achieve the minimum value of the maximum traffic measurement load (i.e., 1120) among all switches, when the value of parameter β is set as 0.3. As a result, the number of iterations for convergence will increase with the value of parameter β decreasing from 0.3. We also conduct the experiments by increasing the number of flows from 200 K to 800 K and the number of controllers from 8 to 16. The results are consistent and the same conclusion can be drawn.

2) Comparison Among DMPS, Csamp and NSPA: In the section, we conduct five sets of simulations to observe the performance comparison among DMPS, cSamp and NSPA.

The first set of simulations observes the performance of the maximum traffic measurement load among all switches with different numbers of flows in Figs. 11–12. The results in both figures show that the maximum traffic measurement load almost linearly increase with the number of flows from 200 K to 800 K

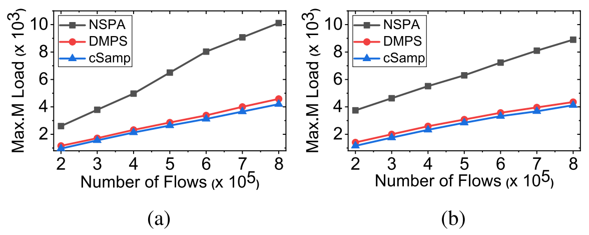


Fig. 11. Max.Measurement Load vs. Number of Flows. *Plot (a):* Topology Fat Tree with 8 Controllers; *Plot (b):* Topology VL2 with 8 Controllers.

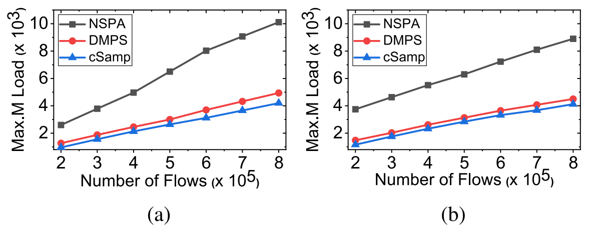


Fig. 12. Max.Measurement Load vs. Number of Flows. *Plot (a):* Topology Fat Tree; *Plot (a):* Topology Fat Tree with 16 Controllers; *Plot (b):* Topology VL2 with 16 Controllers.

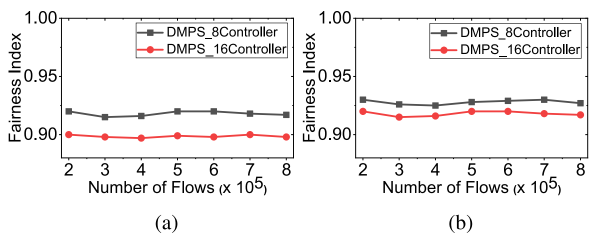


Fig. 13. Fairness Index vs. Number of Flows. *Plot (a):* Topology Fat Tree; *Plot (b):* Topology VL2.

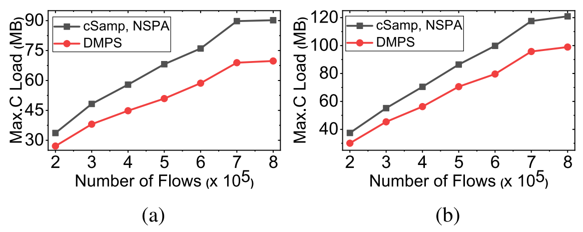


Fig. 14. Max.Communication Load vs. Number of Flows. *Plot (a):* Topology Fat Tree; *Plot (a):* Topology Fat Tree with 8 Controllers; *Plot (b):* Topology VL2 with 8 Controllers.

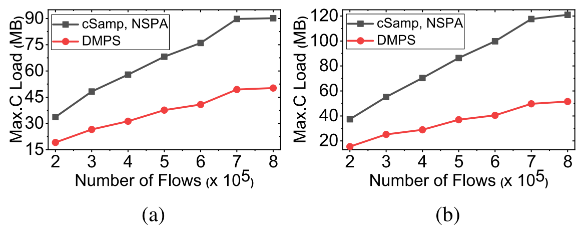


Fig. 15. Max.Communication Load vs. Number of Flows. *Plot (a):* Topology Fat Tree with 16 Controllers; *Plot (b):* Topology VL2 with 16 Controllers.

on both two topologies with different controllers. Specifically, for the topology Fat Tree with 8 controllers depicted in plot (a) of Fig. 11, the maximum traffic measurement loads are 3380, 8030 and 3120 with 600 K flows, respectively, when the algorithms DMPS, NSPA and cSamp are applied for the traffic measurement load distribution. Moreover, for the topology Fat Tree with 16 controllers depicted in plot (a) of Fig. 12, the maximum traffic measurement loads are 3700, 8030 and 3120 with 600 K flows, respectively, when the algorithms DMPS, NSPA and cSamp are applied for the traffic measurement load distribution. That is, our proposed DMPS algorithm can achieve a slightly ($\approx$10% on average) worse performance compared with cSamp in terms of the traffic measurement load balance. Since the algorithm NSPA assigns a sampling probability for each switch, our proposed DMPS algorithm can achieve a much better (at least 100% on average) performance compared with NSPA in terms of the traffic measurement load balance.

The second set of simulations observes the fairness index achieved by our DMPS algorithm under the different numbers of flows. Fig. 13 shows that the fairness index under a certain number of controllers can be maintained at almost the same value, regardless of the number of flows. Specifically, the values of the fairness index are all about 0.92 and 0.9, when the numbers of controllers are 8 and 16 in topology Fat Tree with different number of flows depicted in the plot (a). Moreover, for the plot (b) of Fig. 13, the values of the fairness index are all about 0.93 and 0.92, when the numbers of controllers are 8 and 16 in topology VL2 with different number of flows. We find that the value of the fairness index increases with the number of controllers decreasing. The reason is that the algorithm running on each controller can only make its traffic measurement load distribution strategy based on the traffic information collected by itself and its neighbours. What's more, the number of switches managed by each controller decreases with the number of controllers, leading to the traffic information collected by each controller decreasing.

The third set of simulations observes the maximum communication load among controllers when deploying cSamp, NSPA and DMPS on them. We vary the number of flows from 200 K to 800 K. The results under different topologies with 8 controllers are shown in Fig. 14. Moreover, Fig. 15 shows the result under different topologies with 16 controllers. The results in both figures show that DMPS is the most lightweight algorithm in terms of the communication load between two controllers. For instance, from the plot (a) of Fig. 14, we find that the maximum communication load of our DMPS algorithm is reduced by 25% at least, compared to the cSamp and NSPA under the topology Fat Tree with 8 controllers. For the plot (b) of Fig. 14, the maximum communication load of our DMPS algorithm is reduced by 20% at least, compared to the cSamp and NSPA under the topology VL2 with 16 controllers.

The fourth set of simulations observes the running time of cSamp, NSPA and DMPS, shown in Fig. 16. The results show that DMPS is faster than cSamp but slightly slower than NSPA. However, all algorithms can be executed in minutes. Moreover, we find that the running time of DMPS algorithm only depends on the number of route path in the network, i.e., the scale of the network, which makes our DMPS applicable in practical scenarios.

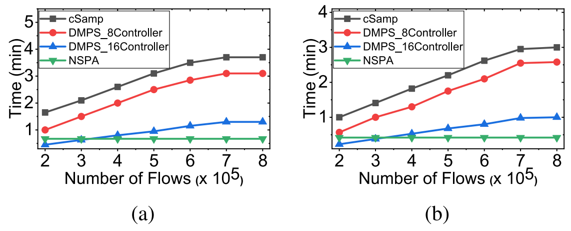


Fig. 16. Running Time vs. Number of Flows. *Plot (a):* Topology Fat Tree; *Plot (b):* Topology VL2.

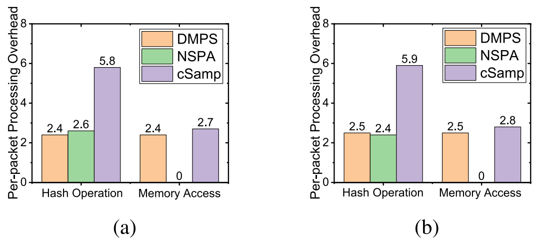


Fig. 17. Per-packet Processing Overhead vs. Operation. *Plot (a):* Topology Fat Tree; *Plot (b):* Topology VL2.

The last set of simulations observes the per-packet processing overhead in the data plane of cSamp, NSPA and DMPS. Fig. 17 shows the per-packet processing overheads of different algorithms on topology Fat Tree and VL2, respectively. Per-packet processing overhead mainly includes hash operations and memory accesses. By the plot (a) of Fig. 17, we observe that cSamp requiresthe 2.3 times numbers of per-packet hash operations as NSPA on topology Fat Tree. Similarly, the number is 2.2 on topology VL. For memory access, all algorithms have very limited number of memory accesses per packet, meaning that packet-processing in the data plane is lightweight.

From the simulation results in Figs. 7–17, we can make the following four conclusions. First, by Figs. 11 and 15, NSPA achieves a slightly worse but comparable performance (≈10% on average) in terms of the traffic measurement load balancing. Second, Figs. 14 and 15 show that our NSPA algorithm can reduce the communication load among controllers drastically, which makes our solution more practical in applications. Third, Fig. 16 shows that our NSPA algorithm can achieve lower time complexity compared to cSamp. Last, Fig. 17 shows that per-packet processing overhead on each switch is lightweight for traffic measurement.

VII. Conclusion

This paper studies how to perform collaborative traffic measurement in multi-controller SDNs. Our solution proposes a dedicate lightweight packet processing design in the data plane to support the collaborative traffic measurement function and an efficient measurement probability assignment strategy in the control plane to improve performance of the solution. To the best of our knowledge, this paper is the first on collaborative traffic measurement in multi-controller SDNs, We prove the convergence of the strategy and conduct extensive simulations to evaluate the performance of the proposed solution with the baselines that are modified from the most related works. The results demonstrate that the proposed solution outperforms baselines in terms of several metrics including measurement load balance and communication load.

References

[1] S. Jain et al., “B4: Experience with a globally-deployed software defined wan,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[2] B. Leng, L. Huang, C. Qiao, H. Xu, and X. Wang, “FTRS: A mechanism for reducing flow table entries in software defined networks,” *Comput. Netw.*, vol. 122, pp. 1–15, 2017.

[3] H. Wang, H. Xu, C. Qian, J. Ge, J. Liu, and H. Huang, “PrePass: Load balancing with data plane resource constraints using commodity sdn switches,” *Comput. Netw.*, vol. 178, 2020, Art. no. 107339.

[4] A. Wang, Y. Guo, F. Hao, T. Lakshman, and S. Chen, “Scotch: Elastically scaling up SDN control-plane using vSwitch based overlay,” in *Proc. 10th ACM Int. Conf. Emerg. Netw. Experiments Technol.*, 2014, pp. 403–414.

[5] H. Xu, H. Huang, S. Chen, and G. Zhao, “Scalable software-defined networking through hybrid switching,” in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[6] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, “AF-QCN: Approximate fairness with quantized congestion notification for multi-tenanted data centers,” in *Proc. 18th IEEE Symp. High Perform. Interconnects*, 2010, pp. 58–65.

[7] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Comput. Secur.*, vol. 28, no. 1/2, pp. 18–28, 2009.

[8] H. Wang, C. Ma, O. O. Odegbile, S. Chen, and J.-K. Peir, “Randomized error removal for online spread estimation in data streaming,” *Proc. VLDB Endowment*, vol. 14, no. 6, pp. 1040–1052, 2021.

[9] Q. Yan, F. R. Yu, Q. Gong, and J. Li, “Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: A survey, some research issues, and challenges,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, Jan.–Mar. 2016.

[10] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 219–230, 2004.

[11] H. Wang, C. Ma, O. O. Odegbile, S. Chen, and J.-K. Peir, “Randomized error removal for online spread estimation in high-speed networks,” *IEEE/ACM Trans. Netw.*, vol. 31, no. 2, pp. 558–573, Apr. 2023.

[12] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic engineering in software defined networks,” in *Proc. IEEE INFOCOM*, 2013, pp. 2211–2219.

[13] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling Flow Management for High-Performance Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, 2011.

[14] J. Sommers, P. Barford, N. Duffield, and A. Ron, “Accurate and efficient SLA compliance monitoring,” in *Proc. Conf. Appl. Technol. Architectures Protoc. Comput. Commun.*, 2007, pp. 109–120.

[15] 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, 2005.

[16] G. Cormode and S. Muthukrishnan, “What’s new: Finding significant differences in network data streams,” *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1219–1232, Dec. 2005.

[17] 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.

[18] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen, “CSAMP: A system for network-wide flow monitoring,” in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 233–246.

[19] Y. Yu, C. Qian, and X. Li, “Distributed and collaborative traffic monitoring in software defined networks,” in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 85–90.

[20] H. Xu, S. Chen, Q. Ma, and L. Huang, “Lightweight flow distribution for collaborative traffic measurement in software defined networks,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1108–1116.

[21] Y. Cui, S. Dong, and W. Liu, “Feature selection algorithm based on correlation between muti metric network traffic flow features.,” *Int. Arab J. Inf. Technol.*, vol. 14, no. 3, pp. 362–371, 2017.

[22] 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, 2018.

[23] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, vol. 3, 2010, pp. 10–5555.

[24] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks," in *Proc. USENIX Conf. Operating Syst. Des. Implementation*, 2010, pp. 1–6.

[25] P. Berde et al., "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.

[26] D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 256–281, Jan.–Mar. 2021.

[27] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A message exchange protocol for software defined networks (SDNs) across multiple domains," Internet Draft, *Internet Eng. Task Force*, Jun. 2012. [Online]. Available: http://tools.ietf.org/id/draft-yin-sdn-sdni-00.txt

[28] P. Lin, J. Bi, and Y. Wang, "WEBridge: West–east bridge for distributed heterogeneous SDN noses peering," *Secur. Commun. Netw.*, vol. 8, no. 10, pp. 1926–1942, 2015.

[29] P. N. D. Bukh, *The Art of Computer Systems Performance Analysis, Techniques for Experimental Design, Measurement, Simulation and Modeling*. Noida, India: Wiley, 1992.

[30] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 27–38.

[31] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 657–665.

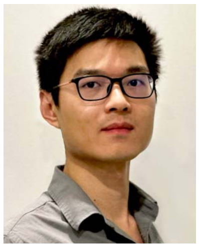
[32] "Openvswitch,", 2017. Accessed: Sep. 1, 2017, [Online]. Available: http://openvswitch.org/

[33] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[34] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 133–145, 2002.

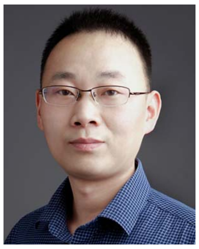
Da Yao received the B.S. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2013. He is currently working toward the Ph.D. degree in computer science with the University of Science and Technology of China, Hefei, China. His main research interests include software-defined networks and traffic measurement.

Qianpiao Ma 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, Hefei, China, in 2014 and 2022, respectively. He is currently a Postdoctoral Researcher with Purple Mountain Laboratories, Nanjing, China. His primary research interests include federated learning, mobile edge computing and distributed machine learning.



Haibo Wang received the Ph.D. degree in computer science from the University of Florida, Gainesville, FL, USA, the B.E. and Masters degrees in computer science from the University of Science and Technology of China, Hefei, China, in 2016 and 2019, respectively. He is currently an Assistant Professor with the Department of Computer Science, University of Kentucky, Lexington, KY, USA. His main research interests include internet traffic measurement, big data analytics, software defined networks, and Internet of Things. He was the recipient of the IEEE ICNP2021 best paper award.

Min Chen (Graduate Student Member, IEEE) received the B.S. degree in software engineering from the Xi'an Jiaotong University, Xi'an, China, in 2018. He is currently working toward the Ph.D. degree with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include mobile edge computing, distributed machine learning, and cloud 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, Hefei, China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored more than 100 papers in famous journals and conferences, which include the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Infocom and ICNP. He has also held more than 30 patents. His main research interests include software defined networks, edge computing and Internet of Thing. He was the recipient of 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.