Computer Networks 178 (2020) 107339

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

PrePass: Load balancing with data plane resource constraints using commodity SDN switches

Haibo Wang[a], Hongli Xu[a,*], Chen Qian[b], Juncheng Ge[c], Jianchun Liu[c], He Huang[d,*]

[a] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
[b] Department of Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064, USA
[c] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
[d] School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu 215006, China

ARTICLE INFO

Keywords:
Software defined networks
Load balancing
Switch resource constraint
Rounding

ABSTRACT

Software Defined Networks (SDN) enables the network control logic to be designed and operated on a global network view by decoupling the control plane from the data plane. Due to versatility and universality of network applications, network layer load balancing is crucial to ensure operational efficiency in a network with a variety of workloads. However, limited resources (*e.g.*, TCAM and computing capacity) on SDN switches bring critical challenges for load balancing. On one hand, though some solutions satisfy resource constraints, these methods do not work well especially for network asymmetry and traffic dynamics. On the other hand, most previous works achieve load balancing with additional hardware or software resources, which increase the system cost and limit the applicability. Thus, this paper tries to deal with the following challenge: *how to achieve load balancing without additional device and software on commodity switches while dealing with network/traffic uncertainties*? To this end, we design and implement PrePass, which uses wildcard entries for some aggregate flows to satisfy the flow table size constraint, and performs reactive routing for newly arrived flows to achieve load balancing even with network/traffic uncertainties. We define the problem of load balancing with flow table size constraint, and prove its NP-hardness. We then present an efficient algorithm based on randomized rounding, and analyze that our algorithm can achieve constant bi-criteria approximation under most practical situations. To make our problem more robust, an extended version without traffic size knowledge is also studied. We implement PrePass on a real SDN testbed. The experimental results and extensive simulation results show that our proposed method can satisfy different resource constraints on switches, and only increase the link load ratio by about 5%-10% compared with per-flow routing scheme under various traffic scenarios.

1. Introduction

Network layer load balancing is an important technique that uses routing decisions to avoid potential congestion on certain links and increase the network bandwidth utilization. Load balancing is important for various types of networks including ISP, data centers [1], WAN [2], and enterprise networks [3]. In particular, modern networks should support an increasingly diverse set of workloads, ranging from small latency-sensitive flows (*e.g.*, search) to bandwidth-hungry large flows (*e.g.*, VM migration [4] or video [5]). To better serve a diversity of flows, load balancing is crucial to ensure operational efficiency and suitable application performance.

As a recent trend, Software Defined Networking (SDN) has been widely used in modern networks. In a typical SDN, the controller monitors the network and determines the forwarding paths of traffic flows. The switches carry out different operations (*e.g.*, forwarding or dropping) for flows/packets based on the rules installed by the controller. Since the controller is able to implement centralized and reactive control for each flow through the header packet reporting mechanism [6], the controller can manage the network in a fine-grained manner, which help to improve the network bandwidth utilization compared with traditional network load balancing [2]. However, limited resources on SDN switches pose additional difficulty for load balancing. Specifically, two types of resources are of major concerns. **(1) Memory resource.** Since Ternary Content Addressable Memory (TCAM) is expensive and power hungry, the size of a TCAM-based flow table is often limited (*e.g.*, 1500 entries on HP 5406zl switch [3]). **(2) Computing resource.** Switches have special hardware for packet forwarding processing. In addition, as specified in the OpenFlow standard [6], each OpenFlow capable switch has a software implemented OpenFlow agent for basic functions like

* Corresponding authors.
E-mail addresses: xuhongli@ustc.edu.cn (H. Xu), huangh@suda.edu.cn (H. Huang).

https://doi.org/10.1016/j.comnet.2020.107339
Received 9 April 2020; Received in revised form 8 May 2020; Accepted 24 May 2020
Available online 28 May 2020

Table 1
Comparison of existing load balancing schemes.

Schemes	Handle dynamics?	Satisfy FTS constraint?	Need hard/software?	Switch computing overhead
ECMP [8]	No	Yes	No	Low
WCMP [10]	Partially			
DRB [12]				
RBDP [16]				
RLJD [17]	Yes	No	No	Low
Hedera [1]				
DevoFlow [3]	Yes	Yes	Yes	Low
DRILL [18]				
HS [14]				
LocalFlow [19]	Yes	Yes	Yes	High
HULA [20]				
Presto[1] [15]				
CONGA [11]				
LetFlow [21]				
PrePass	**Yes**	**Yes**	**No**	**Low**

link layer discovery, which typically runs on a low-end CPU with limited processing power [7]. *These resource constraints certainly limit how individual flows can be controlled in the network for best load balancing.*

A natural and common way for load balancing is Equal-Cost Multi-Path (ECMP) based multi-path forwarding [8,9]. This mechanism distributes the traffic of different flows on multiple equal-cost paths randomly, such that the load among all links can be balanced. Under symmetric network topology, ECMP can achieve satisfactory load balancing effect. However, it may perform poorly for dynamic flow traffic, which is common in practice. To deal with this weakness, the weighted multiple paths, called WCMP [10], is designed. It assigns different weights for paths based on link load distribution with the objective of load balancing. Therefore, WCMP performs better in asymmetric topology compared with ECMP. However, these solutions of multi-path forwarding have three main disadvantages. First, it may perform poorly in asymmetric topologies, which are common in today's networks due to heterogeneous network components or link/device failures [10,11]. Second, since the multi-path forwarding is usually implemented based on flow hashing, it may cause link congestion when hash collisions occur [1,12,13]. Third, these methods often route flows based on the traffic prediction. Due to traffic uncertainties, it may still lead to load imbalance if without immediate flow rerouting.

Many SDN-based load balancing solutions have been proposed to conquer the disadvantages of ECMP-like multi-path forwarding. Most, if not all, of them may incur problems of resource constraints on commodity switches. To deal with the resource constraints, they typically require additional hardware or software to increase the flow table size (FTS) and/or computing capacity of the SDN data plane for load balancing. As a result, they all increase the network deployment cost and complexity.

- *Methods constrained by limited memory resource:* Some flow-level scheduling algorithms perform fine-grained flow management to achieve the load balancingdo without considering flow table size constraint on switches, such as Hedera [1]. To deal with the limited flow table size, the controller pre-deploys default paths (such as ECMP) for all flows by setting up wildcard rules on switches, and then reroutes some elephant flows for better performance by installing per-flow rules on switches, such as DevoFlow [3], Planck [13], and HS [14]. All flows at first will follow the default paths, then the elephant flows will follow the per-flow rules when the per-flow rules are set up. They all require the traffic statistics information of each individual (or elephant) flow for rerouting and load balancing. However, additional hardware/software is required to obtain traffic statistics of individual flows, because those information for the flows through default paths are unavailable only through the flow table. For example, HS [14] mirrors flow traffics to servers for traffic analysis and DevoFlow [3] requires additional functions in the action part of wildcard rules for traffic measurement.
- *Methods constrained by limited computing resource:* Some methods are proposed to achieve load balancing via flow-level or even subflow-level control. For example, Presto [15] implements the flowcell (fixed-size units) scheduling for load balancing. However, since commodity switches have no such computing resource, Presto requires deploying one virtual switch (vswitch) for every physical switch. It is because a vswitch has more powerful processing capacity compared with a physical switch [7]. Meanwhile, subflow-level control also requires massive flow entries, which conflict with the memory resource constraint on switches.

Many previous load balancing methods require additional hardware or software, which brings plenty of weaknesses for network deployment and applications. First, when additional hardware is required, it increases the system cost and limits the applicability. For example, Presto requires to deploy one vswitch for each physical switch, which leads to a higher deployment cost and worse scalability. Second, many current commodity switches (*e.g.*, HP 5460zl [3]) do not support the required software programs, such as flowlet control. Even if future switches support these functions, it will introduce huge cost for switch update and replacement, which may be unnecessary. We summarize existing load balancing solutions in Table 1.

This paper tries to answer the following question: *how to perform load balancing without additional device or software on commodity switches, while conquering network asymmetry and traffic dynamics/uncertainties*? Our important observation is that load balancing can be achieved by controlling only a part ($< 50\%$) of all flows in the network and let the other flows follow aggregate paths deployed in advance, such as ECMP paths. In this work, we design and implement PrePass for load balancing. PrePass periodically computes and installs forwarding rules (*i.e.*, proactive routing) for a part of aggregated flows (not all flows) in advance while considering the flow table size constraint and flows' spatial distribution. The remaining flows will be assigned route paths by the controller reactively. We note that our scheme is different from the previous default path scheme (*e.g.*, DevoFlow [3] and HS [14]), in which all flows will follow default paths and some of them will be rerouted when congestion occurs. PrePass has some significant advantages:

1. PrePass installs wildcard rules for some (not all) flows with switch-switch granularity (explained in Section 2.4) in advance, which helps to reduce the controller-switch interaction overhead. These forwarding rules are determined by the network traffic and the flow table size constraint.
2. PrePass installs forwarding rules with switch-host granularity (explained in Section 2.4) under the reactive scheme rather than with 5-tuple granularity, which can largely reduce flow entry consumption and avoid frequent controller-switch interaction overhead. As

the controller performs reactive routing for other flows, the routing performance (*e.g.*, load balancing or network throughput) will still be efficient even with traffic dynamics, which will be validated by experiments in Section 6.

3. Our method only requires that each SDN switch performs the normal matching-and-forwarding operations, without additional control or computing requirements, which is compatible with commodity switches.

In this paper, we design and implement PrePass for load balancing. Specifically, we define the problem of load balancing with flow table size constraint (LB-FTS), and prove its NP-hardness. A rounding-based algorithm is designed and the analysis shows that the proposed algorithm can achieve the constant bi-criteria approximation under most practical situations. This paper also discusses some practical issues to enhance the practicality of PrePass. We implement PrePass on our SDN testbed. The experimental results and extensive simulation results show that PrePass can satisfy the different resource constraints on switches without additional hardware/software, while only increasing a little link load ratio (about 5%-10%) compared with a routing scheme using unlimited data plane resources, under various and dynamic traffic scenarios.

2. Background and motivation

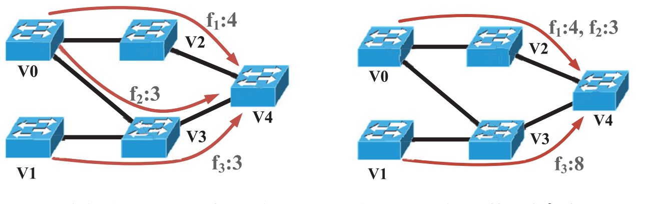
The purpose of this paper is to design a simple and efficient load balancing scheme that conforms to the flow table size and computing capacity constraints on commodity switches. This section presents the key insights underlying our design.

2.1. Load balancing with FTS constraint

Due to the high cost and power consumption of TCAM, commodity switches can only support flow tables with limited size[1] (usually less than 5000 flow entries on commodity switches [22], *e.g.*, 1500 on HP 5406zl switch [3]), which becomes the bottleneck of routing performance, especially in large-scale networks. For example, even in a moderate-size network [23], the number of flows may reach 10^6 [22], which is usually far more than the flow table size. Thus, it is impossible to install a rule for each individual flow. One intuitive way, like RLJD [17], discards some flows so as to satisfy the flow table size constraint, which will reduce the network throughput and experience quality. To accommodate all flows, some wildcard rules are installed to match more flows. For example, we can perform prefix aggregate routing instead of per-flow routing, where flows with the same address prefix share the same path [3,13,14]. Since many flows match one wildcard flow entry and/or follow a single path, it may result in load imbalance. Moreover, the controller cannot obtain traffic statistics information of each individual flow. Thus, it is difficult to achieve load balancing through flow rerouting.

An SDN switch cannot support all flows with per-flow rules for the following two reasons. (1) TCAMs consume lots of ASIC space and power. Specifically, an OpenFlow rule is described by 10 header fields, which costs total 288 bits [6], while an Ethernet forwarding descriptor is 60 bits, thus OpenFlow entries use more states than Ethernet forwarding entries and are impractical to support large quantities of flow entries due to the limited TCAM size on commodity switches. (2) It takes an unacceptably long time to collect statistics for a large flow table. For example, the statistics-pulling latency for the 5406zl switch is less than 1 s when the flow table has fewer than 5600 entries [3]. But this can be increased to 2.5 s if there are 13 K flow table entries [3], which is too long for some practical application flow scheduling [1].

[1] Presto requires extra soft edge (*i.e.*, vSwitch and hypervisor) [15].



(a) Original Route Configuration (b) Dynamic traffic of f_3 from 3 to 8

Fig. 1. Problems caused by traffic dynamics.

2.2. Switch computing capacity constraint

The key characteristic for an SDN is to separate data plane and control plane, and let different parts perform their own functions. The core function of an SDN switch is to forward packets, leaving the control plane in charge of complex decision and control functions. Since most commodity switches run on a low-end CPU, they only have finite computing capacity, which should be mainly responsible for flow entry setup and statistics collection, etc. The testing results in Curtis et al. [3] have shown that the 5406zl switch can complete only 275 flow setups per second even without any traffic load. Due to limited computing capacity, the statistics-pulling interferes with flow setup. When traffic statistics are pulled once a second, collecting less than 4500 counters, the 5406zl switch can only install fewer than 150 flows per second. In other words, statistics pulling will significantly degrade the entry setup functions (from 275 to 150 flow setups per second). Thus, to make SDN switch work efficiently, we expect less extra computing overhead on each switch other than necessary operations, *e.g.*, flow entry setup and statistics collection.

2.3. Network uncertainties

Modern networks are filled with the following two main uncertainties. (1) Asymmetric topologies are common in practical situations due to link failures and heterogeneous switching equipment (*e.g.*, a different number of ports, forwarding speed, *etc.*). For instance, data centers may experience frequent link cuts (40.8 times on average per day caused by protocol issues like UniDirectional Link Detection (UDLD) [24] and device issues in a production data center [24]). (2) Traffic dynamics also contribute to network uncertainty. Previous studies [25] show that traffic in production data centers is highly dynamic, and the link may be heavy-loaded once a few elephant flows burst. An example of route configuration is shown in Fig. 1(a), in which all links have capacity of 10, and flows f_2 and f_3 share the same link V3 → V4. Under this configuration, all links are congestion-free. When flow f_3 changes its traffic intensity from 3 to 8, as shown in Fig. 1(b), flows f_1 and f_2 have to choose path V0 → V2 → V4 to avoid link congestion and achieve load balancing. Through the example, we can see that the load balancing routing scheme is affected by the traffic dynamics and we can change the routing paths of partial flows (flow f_2) to achieve load balancing.

2.4. Load balancing with different flow granularities

To provide qualified services for load balancing and satisfy the resource constraints, like FTS and computing capacities, on the switches, many load balancing solutions adopt different granularities of flow aggregation to save the resource consumption. For example, the default-path scheme distinguishes and schedules flows using their egress switches [26]. For each routing path with different granularities, the flow entries on switches may be wildcard rules, which is essentially one or multiple mask of bits that indicates which parts of a packet header available for matching. We stress that its matching speed is very fast, e.g., 1 billion packet per second [27], and its flow entry consumption is the same, i.e., one. Because TCAM supports search using three different inputs: 0, 1 and *X*. The "*X*" input, which is often referred to as a

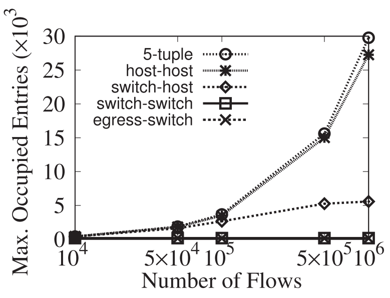


Fig. 2. Maximum occupied flow entries on each switch vs. number of flows.

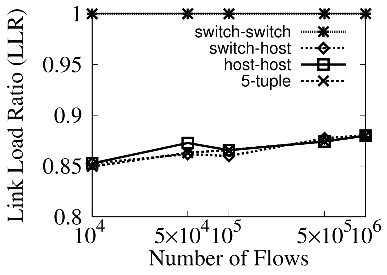


Fig. 3. Link load ratio (normalized to host-host scheme) vs. number of flows.

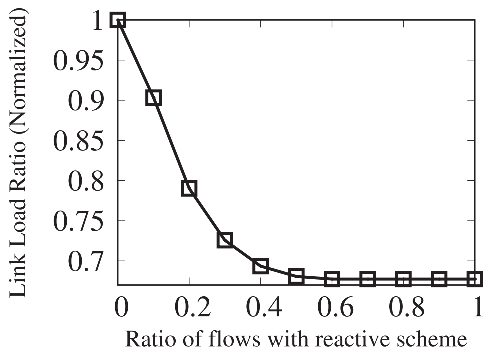
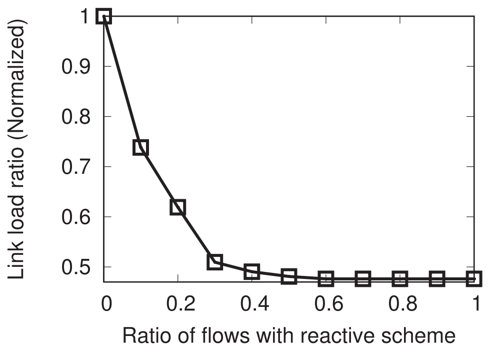



Fig. 4. Ratio of flows with reactive rules with switch-host granularity vs. Link Load Ratio (normalized to 100% switch-switch-granularity load balancing scheme). *top plot*: Data Center Topology; *bottom plot*: Campus Topology.

“wildcard” state, enables TCAM to perform routing policy for different granularities with the one flow entry. Here we list some typical granularities for load balancing in the following.

- egress-switch based: flows with the same egress switch will be regarded as one flow and share one flow entry on a switch [26].
- switch-switch: also called OD-pair (origin-destination switch pair [28]). Flows are distinct if they possess different ingress or egress switches.
- switch-host: flows are distinguished by the source ingress switch and destination host. Note that multiple hosts are connected to the egress switches.
- host-host: all packets with the same source host and destination host are considered as one flow. A switch-host or switch-switch path may contain multiple host-host paths if multiple hosts are connected to the switch.
- 5-tuple: the most fine-grained granularity for load balancing, where flows with different 5-tuples will be arranged separately. A 5-tuple in TCP/IP connection refers to a set of five different values including a source IP address/port number, destination IP address/port number and the protocol in use.

To illustrate the resource consumption and effect of load balancing with different flow granularities, we conduct simulations on a large fat-tree network with 180 switches and 2520 hosts, to evaluate the consumption of flow entries and link load ratio (also called LLR[2]). We adopt the greedy algorithm for load balancing where each new arrival flow will be forwarded to the least light-loaded path. Moreover, we change the number of flows with 5-tuple granularity from 10K/min to 1M/min in the simulation, where 1M flows are practical in today’s data centers [23] and 1min is the default idle time for a flow entry in OpenFlow [29]. The results are shown in Figs. 2 and 3. Fig. 2 shows that, the more fine-grained flow management the routing scheme adopts, the more flow entries are consumed. For instance, the switch-host-granularity routing scheme satisfies the FTS constraint on commodity switches, while the more fine-grained routing schemes (routing schemes with host-host granularity or 5-tuple granularity) may violate the FTS constraint. Fig. 3 shows that the switch-switch-granularity routing scheme and the egress-switch-based routing scheme cannot achieve low LLR like 5-tuple-granularity scheme, increasing 20% compared with other three more fine-grained routing schemes. LLR of egress-switch-granularity routing scheme is not shown in figure because it is too large, increasing 20% compared with other three more fine-grained routing schemes. Therefore, switch-host-granularity routing scheme may be a feasible scheme because of its low LLR like host-host-granularity scheme and 5-tuple-granularity scheme, and its low flow entry consumption like egress-switch-based routing scheme and switch-switch-granularity routing scheme.

2.5. Let partial flows balance the load

In the following, we will show that, even though the controller only installs forwarding rules with switch-switch granularity for partial flows in advance (*i.e.*, proactive scheme) and reactively installs forwarding rules with switch-host granularity for remaining flows (*i.e.*, reactive scheme), the network performance (*e.g.*, LLR) will still be preserved. Meanwhile, this hybrid scheme can largely reduce the controller-switch interaction overhead and the number of flow entry setup for the new-arrival flows under the reactive routing scheme. The network topology can be considered either the asymmetric topology or the symmetric topology. For the generality of our motivation, we adopt two topologies, one for campus networks representing the asymmetric topology and the other for data center networks representing symmetric topology, which will be explained in details in Section 6.3.1. We generate the aggregate switch-switch paths for each flow. Fig. 4 shows that the link load ratio will be reduced when the controller installs switch-host-granularity rules for more flows in the reactive scheme. We find that there is no need to install forwarding rules with switch-host granularity for all flows to

[2] An OpenFlow switch can record the transmitted bytes per port, which is called port_statistics. To calculate the LLR, the controller first collects the port_statistics once the period starts and ends respectively, and then calculates the transmitted bytes through the port during a period. After that, we obtain the link utilization of that port by dividing the time length of the period and the link capacity. Finally, LLR equals to the maximum link utilization of all ports in the network.

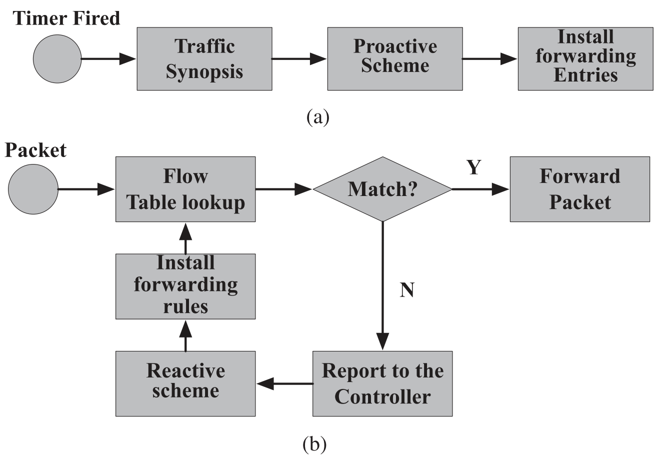


Fig. 5. Illustration of PrePass's workflow. (a) Proactive routing scheme triggered by timer. (b) Reactive routing scheme triggered by new-arrival packets.

achieve better routing performance. Specifically, Fig. 4 shows that the network performance can be almost optimized if the controller only installs switch-host-granularity rules for 30%-40% flows on both topologies. These simulation results motivate us to cope with scalability issues through the proposed hybrid routing scheme.

2.6. System workflow of PrePass

Due to a large quantity of flows in modern networks and the limited size of flow tables, it is impractical to install per-flow rules for every 5-tuple flow. Recall that only partial flows with reactive scheme will also help to achieve better load balancing in Section 2.5. Motivated by the simulation results in Fig. 4, we study the optimal deployment of switch-switch-granularity rules (proactive scheme) and switch-host-granularity rules (reactive scheme) for load balancing. PrePass adopts two routing schemes. (1) Proactive scheme. The controller *periodically* calculates the aggregated routes with switch-switch granularity for a set of flows (also called macroflows [30] in Definition 1). (2) Reactive scheme. The controller will dynamically determine the switch-host-granularity paths for remaining flows to achieve load balancing.

Definition 1 (Macroflow)**.** A macroflow includes multiple flows that share a same path and only cost one (wildcard) flow entry on each switch along the path. Specifically, in this paper all flows from one ingress switch to another egress switch are regarded as a macroflow.

To this end, we separate PrePass's workflow into two main parts: (1) proactive routing scheme triggered by timer, as shown in Fig. 5(a); and (2) reactive routing scheme triggered by new-arrival flows (flows that cannot match current rules on the ingress switch), as shown in Fig. 5(b). For the proactive routing scheme, when a period (*e.g.*, 10min) is fired, PrePass first estimates the traffic synopsis based on the long-term traffic statistics [31]. Then, the controller executes Algorithm 1 to determine how to install wildcard rules (Section 4). Note that proactive routing scheme needs the long-term (not instant) traffic information of macroflows (not flows), that is, this scheme can bear long time delay to collect the coarse-grained (macroflow) information. Thus, existing measurement solutions like sampling and sketches can be applied to obtain traffic information. To be more practical, since the traffic size estimation may be inaccurate, we will discuss how to deal with this case in Sections 4.3.3 and 5.

Now let's introduce the reactive routing scheme in Fig. 5(b). During system running, when a packet arrives at a switch, the switch looks up the flow table. If there is a flow entry matching this packet header, the packet will be processed according to the action field of the matching flow entry. Otherwise, the switch will report the packet header to the controller for path selection. Accordingly, the controller will determine the route path through dynamic flow routing in Section 4.3.2.

Algorithm 1 Rounding-based algorithm for LB-FTS.

```
1: Step 1: Solving the relaxed LB-FTS problem
2: Construct a linear program $LP_1$ based on Eq. (??)
3: Obtain the optimal solution $\widetilde{y}_f^p$, $\widetilde{y}_\gamma^p$ and $\widetilde{z}_\gamma$
4: Step 2: Aggregate routing for selected macroflows
5: Derive an integer solution $\widehat{z}_\gamma$ through randomized rounding method
6: for each macroflow $\gamma \in \Gamma$ do
7:   if $\widehat{z}_\gamma = 1$ then
8:     for each feasible path $p \in \mathcal{P}_\gamma$ do
9:       Compute $\bar{y}_\gamma^p = \frac{\widetilde{y}_\gamma^p}{\widetilde{z}_\gamma}$
       Derive an integer solution $\widehat{y}_\gamma^p$ through randomized rounding method
10:    for each feasible path $p \in \mathcal{P}_\gamma$ do
11:      if $\widehat{y}_\gamma^p = 1$ then
12:        Install wildcard entries on switches along path $p$
```

3. Problem formulation

3.1. Network model

A software defined network (SDN) discussed in this paper consists of a set of SDN switches, $V = \{v_1, v_2, \dots, v_n\}$, with $n = |V|$; and a logically centralized controller. The switches comprising the data plane are responsible for the packet forwarding function. When a flow arrives at a switch, if there is no matched rule for this flow, the controller will determine the route path for this flow in a logically centralized manner. The network topology can be modeled by a graph $G(V, E)$, where E is the set of links connecting two switches in the network.

3.2. Definition of load-balancing with flow table size constraint (LB-FTS)

In an SDN, the flow table size of each switch v_i is denoted by $\beta(v_i)$, and the capacity of each link $e \in E$ is denoted by $c(e)$. We consider a set of macroflows in the network $\Gamma = \{\gamma_1, \dots, \gamma_m\}$, with $m = |\Gamma|$, and the traffic amount of a macroflow γ is denoted by $s(\gamma)$. For example, all flows from one ingress switch to another egress switch can be regarded as a macroflow. Since all flows in a macroflow γ match the same wildcard rule on each switch, the controller can gather the traffic statistics information of this macroflow by the Counter field in the flow entry. We also assume that the controller knows the number of switch-host-granularity aggregate flows $|\gamma|$ in the macroflow γ through long-term traffic statistics for two reasons. (1) The controller can compute the average switch-host-granularity flow size $s(f)$ through statistics collection of forwarding flow entries deployed by the reactive routing scheme, and get the estimated number of these switch-host-granularity aggregate flows as $|\gamma| = \frac{s(\gamma)}{s(f)}$. (2) The number of switch-host-granularity aggregate flows in each macroflow can also be predicted by the state-of-the-art prediction techniques. For example, Peng et al. [31] shows time-advance and high accuracy in terms of traffic forecasting. Thus, *all required information as the input of LB-FTS can be supported by the current commodity switches, without requiring extra resources on control/data planes*. To be more practical, there may be prediction errors of $|\gamma|$ and $s(\gamma)$, which will be discussed in Section 5. Note that our simulation results in Section 6 show that our proposed algorithm can better tolerate these estimation errors compared with other benchmarks described in Section 6 (except the optimal algorithm).

Each macroflow $\gamma \in \Gamma$ can be denoted by $\gamma = \{f_1, \dots, f_{|\gamma|}\}$. Moreover, there is a feasible route path set, denoted by $\mathcal{P}_\gamma$, which is determined based on the management policies and performance objectives (*e.g.*, k shortest paths). For switch-host-granularity flows in the same macroflow, since they usually have the same source and destination, these switch-host-granularity flows usually have the same feasible route paths. Without confusion, we denote $\mathcal{P}_f$ as the feasible route path set for switch-host-granularity flow $f \in \gamma$ and $\mathcal{P}_f$ is same as $\mathcal{P}_\gamma$.

The LB-FTS problem will choose a subset of macroflows Γ' using proactive routing scheme, and deploy a feasible path for each macroflow in Γ'. We note that the flows will be directly forwarded if there exist matched wildcard rules at switches by proactive scheme. Otherwise, the new-arrival packet will be reported to the controller. We expect to (1) reserve enough flow entries for remaining flows using reactive routing scheme and (2) remain much more bandwidth on each link for load balancing. We formulate the problem as:

$$\min \quad \lambda$$

$$\begin{cases} \sum_{p\in\mathcal{P}_f} y_f^p + z_\gamma = 1 & \forall f\in\gamma, \gamma\in\Gamma \\ \sum_{p\in\mathcal{P}_\gamma} y_\gamma^p = z_\gamma & \forall \gamma\in\Gamma \\ \sum_{\substack{f:f\in\gamma,\gamma\in\Gamma\\ p:v\in p, p\in\mathcal{P}_f}} y_f^p + \sum_{\substack{\gamma:\gamma\in\Gamma\\ p:v\in p,p\in\mathcal{P}_\gamma}} y_\gamma^p \le \beta(v) & \forall v\in V \\ \sum_{\substack{\gamma:\gamma\in\Gamma\\ p:e\in p,p\in\mathcal{P}_\gamma}} y_\gamma^p \cdot s(\gamma) \le \lambda\cdot c(e) & \forall e\in E \\ y_f^p, z_\gamma, y_\gamma^p \in \{0,1\} & \forall p\in\mathcal{P}_f, f\in\gamma\gamma\in\Gamma \end{cases} \tag{1}$$

Note that variable y_f^p denotes whether the switch-host-granularity flow $f \in \gamma$ selects the path $p \in \mathcal{P}_f$ or not, and y_γ^p denotes whether the macroflow γ selects the path $p \in \mathcal{P}_\gamma$ as its aggregate path or not. In addition, z_γ represents whether the macroflow γ chooses the proactive scheme or not, and λ is the load balancing factor for the macroflows. The first set of equations means that either macroflow γ chooses the proactive routing scheme or all switch-host-granularity flows in this macroflow choose the reactive scheme. The second set of equations ensures that there is exact one route path selected as the aggregate path once the macroflow chooses proactive scheme. The third set of inequalities means that the total number of flow entries for both proactive and reactive routing schemes doesn't exceed the flow table size of each switch. The fourth set of inequalities ensures the balance of link load for flows choosing the proactive routing scheme. The load balancing factor λ is the maximum link load ratio among all links in the network. Obviously, the larger λ is, the heavier the link load is. The smaller λ is, the larger the throughput is. Our objective is to minimize the load balancing factor, *i.e.*, min λ.

Theorem 1. *The LB-FTS problem is NP-hard.*

Proof. We consider a special case in which each macroflow contains many flows such that all flows have to choose the proactive routing scheme (because of the flow table size constraint). In other words, if the controller deploys per-flow rules for all flows in one macroflow, it may violate the flow table size constraint. Thus all flows in the same macroflow should be regarded as one flow. Under the above circumstance, our LB-FTS problem becomes the unsplittable multi-commodity flow with minimum congestion problem [32], which is NP-hard. Since the special case of the problem is NP-hard, the LB-FTS problem is NP-hard too. □

4. Algorithm description

Due to the NP-hardness, it is difficult to solve the LB-FTS problem in polynomial time. In this section, we propose a rounding-based algorithm for LB-FTS, and then analyze the approximation performance of the proposed algorithm. After that, we give some discussion to enhance the algorithm.

4.1. Algorithm description

In this section, we propose a rounding-based wildcard rule configuration algorithm in PrePass for the LB-FTS problem. For each macroflow (or flow), there is a feasible path set. Actually, the number of feasible paths connecting two terminals could be exponential to the network diameter. To achieve the trade-off between algorithm complexity and network performance, same as [14,17], we only construct some of feasible paths for each switch-host-granularity flow or macroflow. These paths may be the shortest paths between terminals, which can be found by depth-first search. Since Eq. (1) is an integer linear program, it is difficult to be solved directly. The proposed algorithm includes two steps, relaxing the problem and rounding to the integer solution, respectively. In the first step, we construct a linear program as a relaxation of the LB-FTS problem. Specifically, LB-FTS assumes that the traffic of each switch-host-granularity flow or macroflow is splittable and can be routed through multiple feasible paths.

Since the relaxed version of LB-FTS, denoted by LP_1, is a linear program, we can solve it in polynomial time by a linear program solver. Assume that the optimal solution for LP_1 is $\{\widetilde{y}_f^p, \widetilde{y}_\gamma^p, \widetilde{z}_\gamma\}$, and the optimal result is $\widetilde{\lambda}$. Since LP_1 is the relaxation of the LB-FTS problem, $\widetilde{\lambda}$ is the lower-bound result for LB-FTS.

In the second step, we determine how to deploy aggregate paths for some chosen macroflows. For each macroflow γ, we obtain an integer solution $\hat{z}_\gamma$ using the randomized rounding method [33]. More specifically, we set $\hat{z}_\gamma = 1$, which means that an aggregate path will be deployed for macroflow γ, with probability $\widetilde{z}_\gamma$. If $\hat{z}_\gamma = 0$, this means that all switch-host-granularity flows in macroflow γ will be routed with reactive scheme. According to the first set of equations in Eq. (1), we have the following equation after the randomized rounding for $\hat{z}_\gamma$:

$$\sum_{f\in\gamma,p\in\mathcal{P}_f} y_f^p = 1 - \hat{z}_\gamma = \begin{cases} 0, & \text{if } \hat{z}_\gamma = 1 \\ 1, & \text{if } \hat{z}_\gamma = 0 \end{cases} \tag{2}$$

According to the second set of equations in Eq. (1), there is a fractional result $\widetilde{y}_\gamma^p$ for each feasible path $p \in \mathcal{P}_\gamma$, and the sum of all these fractional results is $\sum_{p\in\mathcal{P}_\gamma} y_\gamma^p = \widetilde{z}_\gamma$. Since we will only select one feasible path by randomized rounding, we need to normalize the expectation of these variables. For each feasible path $p \in \mathcal{P}_\gamma^p$, we set $\bar{y}_\gamma^p = \frac{\widetilde{y}_\gamma^p}{\widetilde{z}_\gamma}$. Then, we get the integer solution $\hat{y}_\gamma^p$ through randomized rounding [33]. In particular, $\hat{y}_\gamma^p$ is set as 1 with probability $\bar{y}_\gamma^p$. The PrePass routing algorithm is formally described in Algorithm 1.

4.2. Performance analysis

This section analyzes the approximation performance of PrePass. Assume that the minimum capacity of all the links is denoted by $c_{\min}$. We define a variable α as follow:

$$\alpha = \min\{\beta(v), v\in V; \frac{\widetilde{\lambda}\cdot c_{\min}}{s(\gamma)}, \gamma\in\Gamma\} \tag{3}$$

Under most practical situations, since the flow intensity is usually much less than the link capacity, it follows $\alpha \gg 1$. As PrePass is a randomized algorithm, we analyze the expected resource cost. We give two famous lemmas for probability analysis.

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

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

Link capacity performance We give the following lemma to show the link capacity performance of out algorithm.

Lemma 4. *After the rounding process, the load of each link will not exceed the constraint* $\widetilde{\lambda}\cdot c(e)$ *by a factor of* $\epsilon + 1 = \frac{4\log n}{\alpha} + 3$.

Before analyzing the link resource performance, we define a random variable x_γ^e to denote the traffic amount on link e from macroflow γ.

Definition 2. For each macroflow γ and each link e, a random variable x_γ^e is defined as:

$$x_\gamma^e = \begin{cases} s(\gamma), & \textit{with probability} \sum_{p:e\in p, p\in\mathcal{P}_\gamma} \widetilde{y}_\gamma^p \\ 0, & \textit{otherwise}. \end{cases} \quad (4)$$

According to the definition, $x_{\gamma_1}^e, x_{\gamma_2}^e \ldots$ are mutually independent. The expected link load on link e is:

$$\mathbb{E}\left[\sum_{\gamma\in\Gamma} x_\gamma^e\right] = \sum_{\gamma\in\Gamma}\mathbb{E}[x_\gamma^e] = \sum_{\gamma\in\Gamma}\sum_{p:e\in p,p\in\mathcal{P}_\gamma} \widetilde{y}_\gamma^p \cdot s(\gamma) \le \widetilde{\lambda}\cdot c(e) \quad (5)$$

Combining Eq. (5) and the definition of α in Eq. (3), we have

$$\begin{cases} \frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \in [0,1] \\ \mathbb{E}\left[\sum_{\gamma\in\Gamma} \frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)}\right] \le \alpha. \end{cases} \quad (6)$$

Then, by applying Lemma 2, assume that ϵ is an arbitrary positive value. It follows

$$\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \ge (1+\epsilon)\alpha\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \quad (7)$$

Now, we assume that

$$\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \ge (1+\epsilon)\alpha\right] \le e^{\frac{-\epsilon^2\alpha}{2+\epsilon}} \le \frac{\mathbb{F}}{n^2} \quad (8)$$

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

The solution for Eq. (8) is expressed as:

$$\epsilon \ge \frac{\log\frac{n^2}{\mathbb{F}} + \sqrt{\log^2\frac{n^2}{\mathbb{F}} + 8\alpha\log\frac{n^2}{\mathbb{F}}}}{2\alpha}, \quad n\ge 2 \quad (9)$$

Proof. Set $\mathbb{F} = \frac{1}{n^2}$. Eq. (8) is transformed into:

$$\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \ge (1+\epsilon)\alpha\right] \le \frac{1}{n^4}, \text{where } \epsilon \ge \frac{4\log N}{\alpha}+2$$

By applying Lemma 3, we have,

$$\begin{aligned} &\mathbf{Pr}\left[\bigvee_{e\in E}\sum_{\gamma\in\Gamma}\frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \ge (1+\epsilon)\alpha\right] \\ &\le \sum_{e\in E}\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{x_\gamma^e\cdot\alpha}{\widetilde{\lambda}\cdot c(e)} \ge (1+\epsilon)\alpha\right] \\ &\le n^2\cdot\frac{1}{n^4} = \frac{1}{n^2}, \quad \epsilon \ge \frac{4\log n}{\alpha}+2 \end{aligned} \quad (10)$$

Note that the third inequality holds, because there are at most n^2 links in a network. The approximation factor of our algorithm is $\epsilon + 1 = \frac{4\log n}{\alpha} + 3$. □

Flow table resource performance Lemma 5 guarantees the flow table resource performance.

Lemma 5. *After the rounding process, the number of flow entries on each switch will not exceed the constraint $\beta(v)$ by a factor of $\rho + 2 = \frac{3\log n}{\alpha} + 4$.*

The analysis of Lemma 5 is similar to the analysis of Lemma 4. Therefore, we do not repeat the proof here, which can found in balancing with data plane resource constraints using commodity sdn switches [34].

Approximation factors By Lemmas 4 and 5, we conclude:

Lemma 6. *The traffic load will hardly be violated by a factor of $\frac{4\log n}{\alpha}+3$, and the flow table constraint will not be violated by a factor of $\frac{3\log n}{\alpha}+4$.*

In most practical situations, PrePass can reach almost the constant bi-criteria approximation. For example, let $\widetilde{\lambda}$ and n be 0.4 and 1000. Observing the practical flow traces, the maximum intensity of a macroflow may reach 100 Mbps. In today's networks, the link capacity can be 10 Gbps [9]. The flow table size is usually 4000. Under these circumstances, $\alpha = \frac{\widetilde{\lambda}\cdot c_{\min}}{s(\gamma)}$ will be 40. Thus, the approximation factors for the link capacity and the flow table constraint are 3.7 and 4.5, respectively. In other words, PrePass can achieve the constant bi-criteria approximation for the LB-FTS problem under many practical situations.

Now we discuss the time complexity of the Algorithm 1. Cohen et al. [17] have shown that a set of constant number (k) of paths for each flow are enough for performance optimization compared with all potential polynomial number of feasible paths for each flow. Therefore, for all flows, the maximum number of feasible paths, namely Δ, equals to $k\cdot r$, where r denotes the number of all flows, namely $r = |\Gamma|$. Since the number of variables in the linear program is polynomial value of r and the number of switches (n), it takes polynomial time to solve this linear program. The second step uses randomized rounding for route selection. Specifically, it selects one feasible path as the route path for each macroflow, and then installs flow entries along the route path. Thus, the time complexity is $k\cdot r\cdot\delta = \Delta\cdot\delta$, where δ is the maximum hop number of all feasible paths. As a result, the total time complexity of Algorithm 1 is polynomial of the number of all feasible paths (Δ), the number of switches (n) and the maximum hop number of all feasible paths (δ).

4.3. Discussion

This section provides some discussion to improve the practicability of PrePass.

4.3.1. Port statistics collection

During system running, the controller should master the real-time traffic load on each link to better deal with traffic dynamics. The openflow standard specifies the OFPT_PORT_STATUS interface for port traffic statistics collection [29]. Since each link connects with two ports on different switches, we can collect port traffic statistics only from a subset of switches to reduce the controller overhead. Therefore, we use the minimum set cover algorithm for port traffic statistics collection. Since the number of ports on each switch is usually small (*e.g.*, 24), the overhead (or delay) for port statistics collection is very low (or small). For better trade-off between the statistics accuracy and collection overhead, we trigger the port statistics collection with a fixed period, *e.g.* 5s.

4.3.2. Dynamic flow routing

When a new flow arrives at an SDN switch, this switch reports the packet header to the controller, which will dynamically determine its route path. In this paper, we introduce a simple and efficient routing mechanism, in which the controller chooses the least-congestion switch-host-granularity route path with flow table size constraint for each new-arrival flow. As described in Section 4.3.1, the controller has the knowledge of the link traffic load (or the link load ratio) by collecting the port statistics information at the beginning of each period. Since the flow table of each switch is updated by the controller, the controller can derive the number of occupied flow entries on a switch and determines whether this switch can accommodate this flow or not. For each path p, the congestion of this path is the maximum load ratio of all links on this path. The controller adopts the Dijkstra algorithm [35] to explore the route path with the least congestion for this flow, and installs switch-host-granularity rules on switches along this path. Under this situation, the controller can immediately determine the route path for new-arrival flows, which helps to deal with the dynamics of the traffic.

4.3.3. Dealing with flow estimation deviation

When a flow f arrives at a switch, if there is no matched flow entry and the flow table is saturated, the controller cannot install a flow entry on this switch for flow f. Due to flow estimation deviation, the total number of required flow entries may exceed the flow table size. When it happens, we will aggregate some arrived flows with reactive routing scheme into one flow entry with switch-switch granularity using the wildcard rule. Given a flow f, we determine its corresponding macroflow, denoted by γ, choose a least-congestion path for this macroflow, and update the flow tables on different switches. First, we add a wildcard rule on each switch along this aggregate path. Then, we will remove all flow entries for switch-host-granularity flows in this macroflow on different switches. As a result, the number of required flow entries on switches is reduced.

5. Algorithm description for the extended case

The previous section studies the efficient deployment of aggregate paths for macroflows based on the long-term traffic statistics. To make our problem more robust and generalized, this section studies the case in which the controller has no knowledge of the traffic size of each switch-host-granularity flow/macroflow. In this case, we expect to adjust/control as many switch-host-granularity flows as possible, which contribute to load balancing. For ease of expression, we denote PrePassE as the algorithm for this extended case.

5.1. Definition of load-balancing with maximum adjustable flows (LB-MDF)

Compared with proactive routing scheme, reactive routing scheme (with switch-host-granularity rules) can provide more fine-grained control for each flow. If one flow is forwarded through the reactive scheme, we call this as an adjustable flow. To achieve better load balancing, we expect to remain as many adjustable flows as possible while taking the limited flow table size into account. We formulate the load balancing with maximum adjustable flows (LB-MDF) problem as:

$$\max \sum_{f\in\gamma,\gamma\in\Gamma,p\in\mathcal{P}_f} y_f^p$$

$$\begin{cases} \sum_{p\in\mathcal{P}_f} y_f^p + z_\gamma = 1 & \forall f\in\gamma,\gamma\in\Gamma \\ \sum_{p\in\mathcal{P}_\gamma} y_\gamma^p = z_\gamma & \forall \gamma\in\Gamma \\ \sum_{\substack{f:f\in\gamma,\gamma\in\Gamma\\ p:v\in p,p\in\mathcal{P}_f}} y_f^p + \sum_{\substack{\gamma:\gamma\in\Gamma\\ p:v\in p,p\in\mathcal{P}_\gamma}} y_\gamma^p \le \beta(v) & \forall v\in V \\ y_f^p, z_\gamma, y_\gamma^p \in \{0,1\} & \forall p\in\mathcal{P}_f, f\in\gamma\gamma\in\Gamma \end{cases} \tag{11}$$

The first set of equations means that either macroflow γ chooses the proactive routing scheme or all flows in this macroflow choose the reactive scheme. The second set of equations ensures there is exact one route path selected as the default path once the macroflow chooses the proactive scheme. The third set of inequalities means that the total number of flow entries does not exceed the flow table size constraint on each SDN switch.

Theorem 7. *The LB-MDF problem is NP-hard.*

Proof. We prove the NP-hardness by showing that the 0–1 knapsack problem [36] is a special case of our LB-MDF problem. Considering a special case of LB-MDF, in which the network only has one switch u (or all switches have infinite flow entries except that only one switch has limited flow table size). There are k macroflows $\{\gamma_1,\gamma_2,\dots,\gamma_k\}$, in which each macroflow γ_i includes $|\gamma_i|$ flows. The flow table size on the switch u is $\beta(u)$. We construct a knapsack whose capacity is $\beta(u)-k$, and each item i whose weight and value are $w_i=|\gamma_i|-1$ and $v_i=w_i=|\gamma_i|-1$, respectively. The 0–1 knapsack problem is to maximize the total values of items in the knapsack while the total weight is less than or equal to the knapsack capacity. Assume that the optimal solution of the 0–1 knapsack problem is denoted by $B=\{b_1,b_2,\dots,b_k\}$, where b_i is a binary variable representing whether the item i is selected (b_i=1) or not (b_i=0), and we have $\sum_{i=1}^k b_i\cdot w_i\le\beta(u)-k$. If the item i is selected by the optimal solution in the 0–1 knapsack problem, macroflow γ_i will choose proactive routing, otherwise, reactive routing. Thus the macroflow γ_i will occupy $b_i\cdot w_i+1$ flow entries. The total number of occupied flow entries for all macroflows is $\sum_{i=1}^k(b_i\cdot w_i+1)\le\beta(u)-k+k=\beta(u)$. Thus our constructed example of 0–1 knapsack can satisfy the flow table size constraint. Similar analysis for the objective to maximize the number of adjustable flows. Thus the optimal result B for 0–1 knapsack problem is the optimal result for the special case of LB-MDF problem. Obviously, the 0–1 knapsack problem is NP-hard [36]. Since the special case of our LB-MDF problem is NP-hard, LB-MDF is NP-hard too. □

5.2. Algorithm description

In this section, we propose a rounding-based wildcard rule configuration algorithm called PrePassE for the LB-MDF problem. Since Eq. (11) is an integer linear program, it is difficult to solve it directly, thus we first relax the integer linear program by replace the fourth constraints with $y_f^p, z_\gamma, y_\gamma^p\in[0,1]$. That is, each switch-host-granularity flow is splittable and can be forwarded to several paths. After the relaxation, Eq. (11) becomes a linear program, and we can solve it in polynomial time by a linear program solver. Note that $\{\widetilde{y}_f^p,\widetilde{y}_\gamma^p,\widetilde{z}_\gamma\}$ is the optimal solution for the linear program. Since the linear program is the relaxation of the LB-MDF problem, the optimal result $\widetilde{\theta}=\sum_{f\in\gamma,\gamma\in\Gamma,p\in\mathcal{P}_f}\widetilde{y}_f^p$ is the upper-bound result for LB-MDF.

In the second step, we determine how to deploy an aggregate path for each macroflow. We obtain an integer solution $\widehat{z}_\gamma$ using the rounding method [33]. More specifically, we set $\widehat{z}_\gamma=1$, which means that an aggregate path is deployed for macroflow γ, with the probability $\widetilde{z}_\gamma$. If $\widehat{z}_\gamma=0$, this means that all the switch-host-granularity flows in macroflow γ will be routed with forwarding rules. According to the first set of equations in Eq. (11), we have Eq. 12 after randomized rounding:

$$\sum_{f\in\gamma,p\in\mathcal{P}_f} y_f^p=\begin{cases}0, & \text{if } \widehat{z}_\gamma=1\\ 1, & \text{if } \widehat{z}_\gamma=0\end{cases} \tag{12}$$

According to the second set of equations in Eq. (11), there is a fractional result $\widetilde{y}_\gamma^p$ for each feasible path $p\in\mathcal{P}_\gamma$, and the sum of all these fractional results is: $\sum_{p\in\mathcal{P}_\gamma}y_\gamma^p=\widetilde{z}_\gamma$. Since we need to select one path by randomized rounding, it is required to normalize the expectation of these variables. For each feasible path $p\in\mathcal{P}_\gamma^p$, $\bar{y}_\gamma^p=\frac{\widetilde{y}_\gamma^p}{\widetilde{z}_\gamma}$. Then, we get integer solution $\widehat{y}_\gamma^p$ through randomized rounding [33]. In particular, $\widehat{y}_\gamma^p$ is set as 1 with the probability $\bar{y}_\gamma^p$. PrePassE is formally described in Algorithm 2.

Algorithm 2 Rounding-based algorithm for LB-MDF.

```
1:  Step 1: Solving the relaxed LB-MDF problem
2:  Construct a relaxed linear program based on Eq. (??)
3:  Obtain the optimal solution $\widetilde{y}_f^p$, $\widetilde{y}_\gamma^p$ and $\widetilde{z}_\gamma$
4:  Step 2: Proactive routing for selected macroflows
5:  Derive an integer solution $\widehat{z}_\gamma$ by randomized rounding
6:  for each macroflow $\gamma\in\Gamma$ do
7:    if $\widehat{z}_\gamma = 1$ then
8:      for each feasible path $p\in\mathcal{P}_\gamma$ do
9:        Compute $\bar{y}_\gamma^p=\frac{\widetilde{y}_\gamma^p}{\widetilde{z}_\gamma}$
10:     Derive an integer solution $\widehat{y}_\gamma^p$ by randomized rounding
11:     for each feasible path $p\in\mathcal{P}_\gamma$ do
12:       if $\widehat{y}_\gamma^p = 1$ then
13:         Install wildcard entries on switches along $p$
```

5.3. Performance analysis

In this section, we analyze the approximation performance of PrePassE in terms of the number of adjustable flows and the flow table resource. We first define a variable η as follows:

$$\eta = \min\{\beta(v), v \in V; \widetilde{\theta}/|\gamma|, \gamma \in \Gamma\} \tag{13}$$

Under most practical situations, it follows that $\eta \gg 1$. As PrePassE is a randomized algorithm, we analyze the expected data plane resource cost.

Lemma 8. *The proposed PrePassE algorithm can guarantee that the total number of adjustable flows will not be less than the upper bound of LB-MDF by a factor of* $(1 - 2\sqrt{\frac{\log n}{\eta}})$*,*

Proof. To prove the Lemma 8, we first introduce a widely used formula.

Lemma 9 (Chernoff Low Bound). *Given n independent variables: $x_1, x_2, \ldots, x_n$, where $\forall x_i \in [0, 1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^{n} x_i]$. Then,* $\mathbf{Pr}\left[\sum_{i=1}^{n} x_i \le (1-\xi)\mu\right] \le e^{\frac{-\xi^2\mu}{2}}$*, where ξ is an arbitrarily positive value in [0,1].*

Since all flows in a macroflow adopt the same (proactive/reactive) routing scheme, we define a random variable π_γ to denote the number of adjustable flows in the macroflow γ.

Definition 3. For each macroflow γ, variable π_γ is defined as:

$$\pi_\gamma = \begin{cases} |\gamma|, & \textit{with probability } 1 - z_\gamma \\ 0, & \textit{otherwise.} \end{cases} \tag{14}$$

According to the definition, π_{γ_1}, π_{γ_2} ... are mutually independent. The expected number of ajustable flows are:

$$\begin{aligned} \mathbb{E}\left[\sum_{\gamma\in\Gamma} \pi_\gamma\right] &= \sum_{\gamma\in\Gamma} \mathbb{E}[\pi_\gamma] = \sum_{\gamma\in\Gamma} |\gamma| \cdot (1 - z_\gamma) \\ &= \sum\nolimits_{\gamma\in\Gamma} \sum\nolimits_{p\in \boldsymbol{P}_f, f\in\gamma} \widetilde{y}_f^p = \widetilde{\theta} \end{aligned} \tag{15}$$

Combining Eq. (15) and the definition of η in Eq. (13), we have

$$\begin{cases} \frac{\pi_\gamma \cdot \eta}{\widetilde{\theta}} \in [0, 1] \\ \mathbb{E}\left[\sum_{\gamma\in\Gamma} \frac{\pi_\gamma \cdot \eta}{\widetilde{\theta}}\right] \le \eta. \end{cases} \tag{16}$$

Then, by applying Lemma 9, assume that ξ is an arbitrary positive value. It follows

$$\mathbf{Pr}\left[\sum\nolimits_{\gamma\in\Gamma} \pi_\gamma \cdot \eta/\widetilde{\theta} \le (1-\xi)\eta\right] \le e^{\frac{-\xi^2\eta}{2}} \tag{17}$$

Now, we assume that

$$\mathbf{Pr}\left[\sum\nolimits_{\gamma\in\Gamma} \pi_\gamma \cdot \eta/\widetilde{\theta} \ge (1-\xi)\eta\right] \le e^{\frac{-\xi^2\eta}{2}} \le \boldsymbol{F}/n \tag{18}$$

where $\boldsymbol{F}$ is the function of network-related variables (such as the number of switches *n, etc.*) and $\boldsymbol{F} \to 0$ when the network size grows. The solution for Eq. (18) is expressed as:

$$\xi \ge \sqrt{2\log n - 2\log \boldsymbol{F}/\eta}, \quad n \ge 2 \tag{19}$$

Set $\boldsymbol{F} = \frac{1}{n}$. Eq. (18) is transformed into:

$$\mathbf{Pr}\left[\sum_{\gamma\in\Gamma} \frac{\pi_\gamma \cdot \eta}{\widetilde{\theta}} \le (1-\xi)\eta\right] \le \frac{1}{n^2}, \text{where } \xi \ge 2\sqrt{\frac{\log n}{\eta}}$$

The approximation factor of our algorithm is $1 - \xi = 1 - 2\sqrt{\log n/\eta}$. □

In most practical situations, the approximation factor is constant. In a network with n switches, the number of macroflows is $n(n-1)$. We assume that only 30% of the flows are adjustable and the number of flows in each macroflow obeys poisson distribution. Note that the expected number of flows in macroflow γ is $|\bar{\gamma}|$, and we have $|\gamma| \sim P(|\bar{\gamma}|)$. According to poisson distribution and mathematical computation, $Pr(|\gamma| > 4|\bar{\gamma}|) < 0.0012$, thus we have $\max\{|\gamma|, \gamma \in \Gamma\} \le 4|\bar{\gamma}|$, and η is $0.3 \cdot \frac{1}{4} \cdot n(n-1)$. For example, the number of switches is usually more than 100, and the number of flow entries on each switches is about 4000 [3]. Thus $\eta = \min\{\beta(v), v \in V; \widetilde{\theta}/|\gamma|, \gamma \in \Gamma\} = 750$, and the approximation factor is 0.84. For a large-scale network with 1000 switches, the approximation factor is 0.92. In summary, PrePassE can achieve constant approximation performance in terms of the number of adjustable flows compared with the upper bound of LB-MDF.

Lemma 10. *The proposed PrePassE algorithm can achieve the approximation factor of* $\frac{3\log n}{\eta} + 4$ *for the flow table size constraint.*

The proof is similar to the proof of Lemma 4. Therefore, we omit the detailed proofs of the above lemma here, which can be found in balancing with data plane resource constraints using commodity sdn switches [34]. Moreover, The analysis for Lemma 4 shows that the approximation factor is constant in most practical situations, which is applicable here.

6. Performance evaluation

To demonstrate the feasibility and efficiency of the proposed algorithms, we conduct the experiment and simulation-based evaluation.

6.1. Performance metrics and benchmarks

We have designed PrePass for proactive and reactive routing schemes, respectively. For ease of description, we denote PrePass as the integrated two algorithms. We evaluate PrePass through testbed implementation and large-scale network simulations. We adopt five metrics for performance evaluation. The first metric is the number of adjustable flows (NAF), which refers to the number of flows that can be adjusted/controlled by the controller in the reactive scheme. Maximizing NAF is the objective of PrePassE. The larger NAF means that we can control/adjust more individual flows, which are of benefit to load balancing in general. Since the packet header of each adjustable flow will be reported to the controller, the number of adjustable flows greatly reflects the controller overhead. Since this paper studies the load balancing under strict flow table size constraint by deploying aggregate paths for partial macroflows, the second metric is link load ratio (LLR). LLR is defined as $\max\{f(e)/c(e), e \in E\}$, where $f(e)$ is the traffic load of link e. The smaller LLR means better load balancing effect. Another important data-plane resource is flow entries. We adopt the cumulative distribution function (CDF) of occupied flow entries on all switches as the third metric. When the number of required flow entries exceeds the flow table size, or the link is congested, flows may be dropped. Thus we adopt throughput as the fourth metric, which is defined as the total traffic amount of flows that are successfully forwarded in the network with flow table size constraint. The final metric is flow completion time (FCT), which defined as the lasting time that the flow transmission is finished. FCT is an important metric [37] because some services are delay-sensitive. This section evaluates both the average FCT and the 99th percentile FCT for worst case performance [38]. To evaluate the performance of our proposed algorithms, we adopt four benchmarks:

1. OSPF [35]. Each switch constructs the shortest path to each destination. Therefore, the flow granularity is egress-switch based granularity.
2. ECMP [8]. Each switch constructs multiple (*e.g.*, 3 in our evaluaion) equal-cost paths to each another switch. Therefore, it will first construct multiple equal-cost paths with egress-switch-based granularity. Then, for each packet matching the destination address, it will be for-warded to one of the multiple paths randomly. This kind of randomization is based on packet level. We should note that the ECMP method requires group entries when there are multiple equal-cost paths to the destination in an SDN. We will compare it with our algorithms in terms of the link load ratio and throughput.

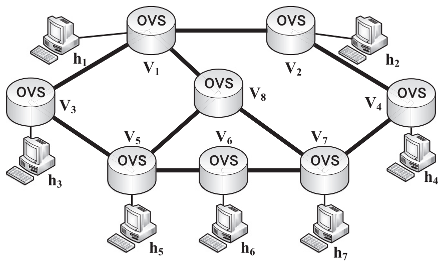


Fig. 6. Topology of the SDN platform.

3. RLJD [17]. In RLJD, flows are forwarded with per-flow rules (5-tuple flows). Since RLJD discards flows if there are not enough flow entries, we will compare it with our algorithm in terms of network throughput. Moreover, we use RLJDM to represent that the flow table size is infinite. That is, each flow will be routed with per-flow rules. We compare RLJDM with our proposed algorithm by evaluating the link load ratio and CDF of occupied flow entries.
4. Presto [15], which applies OVS [39] on the edge switches to enhance the flow table and the processing capacity of edge switches. After that, it will follow the default path with egress-switch-based granularity, which is like OSPF. Though Presto can provide flowcell scheduling in a network, we assume that the controller schedules each flow (not flowcell) on its ingress switch for fairness.

6.2. System implementation and testing

6.2.1. Implementation on the platform

We implement PrePass, OSPF, RLJDM, Presto and ECMP on a small-scale testbed. Our testbed is mainly composed of three parts: a controller, eight virtual switches and seven virtual machines (acting as hosts). Each virtual switch is implemented using the open virtual switch (OVS version 2.4.0) [39], which supports OpenFlow v1.3 standard [6]. Each virtual switch and the connected Kernel-based Virtual Machine (KVM) are implemented on a server with a core i5-3470 processor and 8 GB of RAM. Since the controller will not participate in data forwarding, it is not explicitly shown in Fig. 6. We use Ryu 4.17 [40] as the controller software running on a server with a core i5-3470 processor and 16 GB of RAM.

In our implementation, we generate different quantities of flows on a small-scale topology. Specifically, each virtual machine (or host) is installed with a traffic generator, and requests flows according to Poisson distribution from randomly chosen hosts with different ports. The flow size is synthetic but accords with the heavy-tailed distribution of *data mining workload* [8], where 95% of the traffic amount comes from about 3.6% of elephant flows (more than 10MB). The traffic generator is adopted from [41]. As mentioned in Section 2.4, flow routing using the switch-host granularity will benefit to load balancing in large-scale networks. However, the testbed's topology is of small scale. We thus distinguish a flow using the 5-tuple granularity in the Implementation part so as to generate an adequate number of flows. The flows from a source host will be forwarded to a destination host randomly. To estimate the fabric's load balancing performance, we expect flows to traverse the fabric. Moreover, the proactive routing scheme manages macroflows with the long-term traffic size of these macroflows. But, in practice, each flow in a macroflow may just last for a much shorter time and cause link load fluctuation. Since multiple flows belonging to a macroflow start and finish in different time, it will alleviate this variation. In the experiments, the proactive routing scheme is triggered every 5s, and the link load ratio (LLR) is computed by collecting port statistics. Since the traffic between two hosts attached with the same switch doesn't traverse the fabric, it has no impact on the fabric's load. Then, we deploy one host for each switch and use source IP, destination IP and source port to identify a flow. We also identify a macroflow with the same source IP and destination IP. Then, each host is able to generate different numbers of flows in a network. All the links have the unique link bandwidth, *i.e.*, 1 Gbps.

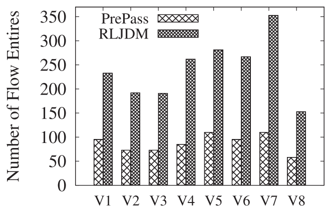


Fig. 7. No. of occupied flow entries on switches.

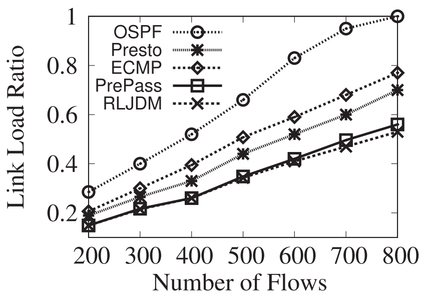


Fig. 8. No. of flows vs. link load ratio (LLR).

Table 2
Number of adjustable flows (NAF) by PrePass.

No. of flows	200	300	400	500	600	700	800
NAF	200	226	213	203	196	195	191

6.2.2. Experimental results

We run three sets of experiments on the SDN platform. In each experiment, we set the flow table size on each switch is 110. In the first testing, we generate 600 flows in the network, and observe the number of required flow entries on all switches. The testing results in Fig. 7 show that the maximum number of occupied flow entries is 110 and 353 for PrePass and RLJDM, respectively. In addition, PrePass satisfies the flow table size constraint on each switch. That is because PrePass deploys aggregate paths for partial macroflows with proactive scheme to reduce the occupied flow entries. Specifically, PrePass can averagely reduce the number of occupied flow entries by $\frac{241-87}{241}$=64% compared with RLJDM. In the second testing, we observe the number of adjustable flows and link load ratio by changing the number of flows in an SDN. The results in Table 2 show that (1) when the number of flows increases (to 300 flows), the number of adjustable flows increases too; and (2) when the number of flows increases (more than 300 flows), the number of adjustable flows decreases. That is because more macroflows will be forwarded through aggregate paths with more and more flows in a network. We observe LLR by changing the number of flows from 200 to 800. In Fig. 8, LLR of PrePass is close to that of RLJDM (within 5%), while PrePass significantly reduces the link load ratio by 21%, 26% and 44% compared with Presto, ECMP, and OSPF, respectively. That is because PrePass deploys aggregate paths with the objective to minimize LLR of aggregate paths and dynamically routes flows with per-flow rules for others.

We observe the FCT performance by generating different numbers of flows on the platform. We do not compare our PrePass algorithm with Presto and RLJDM, for Presto requires additional OVS placed on each ingress switch, and RLJDM requires a large number of flow entries on switches. Similar to PIAS [42], we evaluate the FCT performance of small flows ($<$ 100 KB) and large flows ($>$ 1 MB), respectively, for comprehensive comparison. The FCT results under 600 flows and 1200

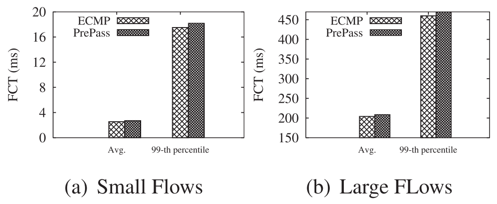


Fig. 9. Performance comparison of FCT under 600 flows.

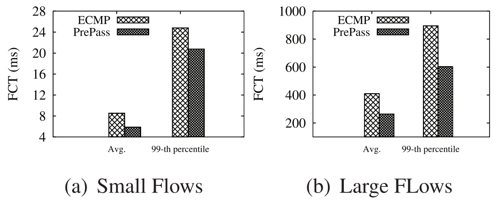


Fig. 10. Performance comparison of FCT under 1200 flows.

flows are shown in Figs. 9 and 10, respectively. As we can see, when there are 600 flows (without congestion), PrePass increases the average FCT and 99th percentile FCT a little (both within 5%) compared with ECMP for both small flows and large flows by Fig. 9. That is because PrePass needs to report Packet-In messages for partial flows to the controller, which will increase the FCT. With more flows in the network, the link load will be increased. As a result, the FCT and the 99th percentile FCT increase for both two algorithms by comparing Figs. 9 and 10. Fig. 10 shows that PrePass performs better than ECMP on both the average FCT and the 99th percentile FCT. Under this case, the link load by ECMP is close to its link capacity. As a result, the impact of queuing delay on the FCT will outweigh that of the controller-switch interaction delay. Since PrePass can achieve lower link load ratio than ECMP, it outperforms ECMP in terms of the FCT performance. For instance, PrePass reduces the average FCT and 99th percentile FCT of large flows by 35% and 32% compared with ECMP, respectively, by Fig. 10(b).

6.3. Simulations

6.3.1. Simulation setting

We select two practical topologies. The first topology is fat-tree [9], denoted by T1, containing 80 switches and 1120 hosts. It has been widely applied in many data centers. The second topology, denoted by T2, is a campus network containing 100 switches and 1000 hosts from [43]. For both topologies, each link has a uniform capacity, 5 Gbps. We conduct extensive simulations with realistic workloads based on the empirical traffic patterns in practical networks. Specifically, we adopt two traffic workloads. The first distribution is derived from traffic traces from a practical data center [11] and represents a large *enterprise workload*. The second distribution is from a large cluster running *data mining* jobs [8]. Both workloads are heavy-tailed: a small fraction of flows contributes most of the traffic amount. In particular, the data mining distribution has a very heavy tail with 95% of the traffic amount from about 3.6% of elephant flows larger than 20 MB, while 20% of the flows account for 80% of the traffic for enterprise workloads. Same as [25], we generate flows between random senders and receivers with varying traffic loads. The flow table size is different for various switches. By default, FTS on each switch is set to a moderate value (*e.g.*, 4 K [3]) in our simulations.

6.3.2. Routing performance and the number of occupied flow entries

We first observe the number of adjustable flows (NAF) by changing the number of flows in a network. Figs. 11 and 12 show that NAF of PrePass increases as the number of flows increases. But when the number of flows increases to 6×10^4, the number of adjustable flows is stable in a certain interval. For example, left plot of Fig. 11 shows that

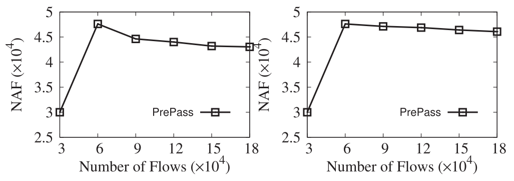


Fig. 11. Number of flows vs. NAF in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

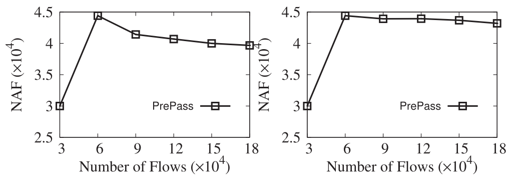


Fig. 12. Number of flows vs. NAF in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

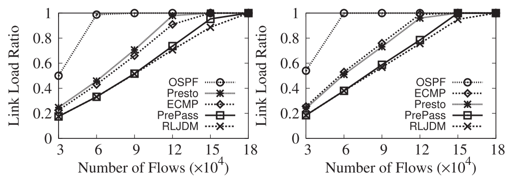


Fig. 13. Number of flows vs. LLR in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

NAF increases from 3×10^4 to 4.7×10^4 when the number of flows increases from 3×10^4 to 6×10^4. After then, NAF remains in the interval $[4.4 \times 10^4, 4.7 \times 10^4]$. The reason is as follow. When there are less flows, *e.g.*, 30 K, the flow table is enough to accommodate all flows with per-flow rules. However, when the number of flows keeps increasing, due to the limited flow table size, more and more flows will pass through the aggregate paths. As a result, NAF is stable in a certain interval. From these two figures, we also conclude that the controller overhead keeps stable with more and more flows in a network.

The second set of simulations studies the link load ratio performance when the number of flows increases. We compare PrePass with OSPF, RLJDM, Presto and ECMP under different numbers of flows ranging from 3×10^4 to 18×10^4. The link load ratio (LLR) results on topology T1 are shown in Fig. 13. We observe that LLR increases with more and more flows for all five algorithms. In comparison, ECMP can reduce LLR by 10% compared with Presto for the enterprise workload, while for the data mining workload, Presto reduces LLR by about 6% compared with ECMP. PrePass can achieve almost the same LLR as RLJDM, especially when the number of flows is less than 9×10^4. For example, when the number of flows is 9×10^4 with the enterprise workload by Fig. 13, OSPF occurs congestion (or LLR is 1), while the link load ratios of PrePass, RLJDM, Presto and ECMP are 0.52, 0.52, 0.72 and 0.66, respectively. As a result, PrePass can reduce LLR by 27.7% and 21.2% compared with Presto and ECMP, respectively. However, simulation results on topology T2 are quite different. ECMP has higher LLR than Presto, RLJD and PrePass. For example, when the number of flows is 6×10^4 with the data mining workload, LLRs of ECMP, Presto, RLJD and PrePass are 0.69, 0.53, 0.39 and 0.39, respectively. That accords with previous researches [15], which have shown that ECMP performs less efficiently on asymmetric topology like topology T2. For both topologies, PrePass achieves close LLR (within 5%) to RLJDM and outperforms Presto and ECMP, even when the number of flows is large (*e.g.*, 150 K) and most of the flows pass through aggregate paths.

The third set of experiments evaluates the cumulative distribution function (CDF) of occupied flow entries on all switches. We compare

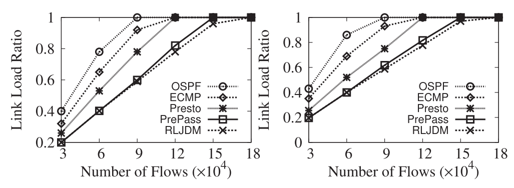


Fig. 14. Number of flows vs. LLR in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

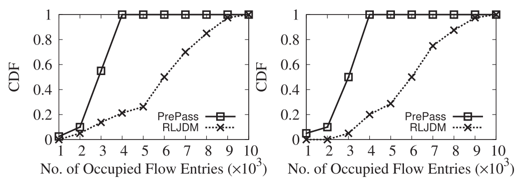


Fig. 15. No. of occupied flow entries vs. CDF in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

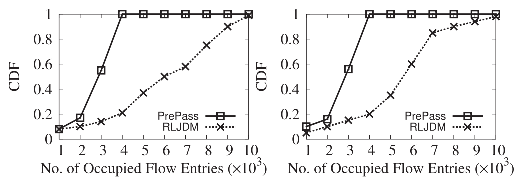


Fig. 16. No. of occupied flow entries vs. CDF in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

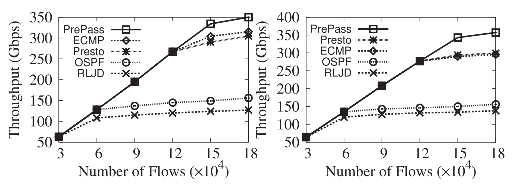


Fig. 17. No. of flows vs. throughput in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

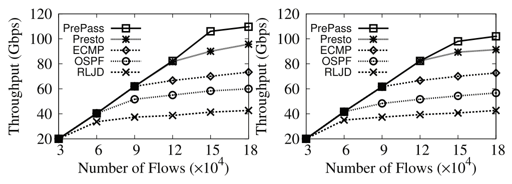


Fig. 18. No. of flows vs. throughput in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

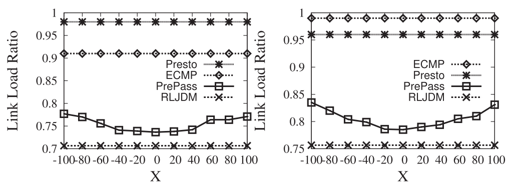


Fig. 19. Prediction error parameter X vs. LLR in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

the number of occupied flow entries for PrePass with RLJDM. As shown in Figs. 15 and 16, PrePass needs no more than 4 K flow entries, while RLJDM uses at most 10K flow entries. Moreover, most of switches need 2 K-4 K entries by PrePass. But for RLJDM, more than 79% of the switches exceed the FTS constraint. That is because PrePass has deployed aggregate paths for partial macroflows, and flows in these macroflows will directly match wildcard entries without consuming extra flow entries. Thus, PrePass can significantly reduce the number of occupied flow entries, and satisfy the FTS constraint on each switch.

The fourth set of simulations observes the network throughput performance when the flow table size is constant (*i.e.*, 4000). The results are shown in Figs. 17 and 18. We observe that the throughput increases as the number of flows increases for all five algorithms. But RLJD and OSPF can accommodate much less traffic compared with Presto, ECMP and PrePass. That is because OSPF has high link load ratio and RLJD is constricted by the flow table size. ECMP has similar throughput as Presto in topology T1, but much less throughput compared with Presto in topology T2. For example, when the number of flows is 15×10^4, ECMP increases throughput by 4.8% compared with Presto in terms of the enterprise workload by Fig. 17, But in topology T2, Presto increases throughput by 27.7% compared with ECMP by Fig. 18(a). Moreover, PrePass achieves much more throughput than other four benchmarks on both topologies and workloads. For example, in Fig. 18 with the enterprise workload, PrePass increases the throughput by 22.2%, 52.7%, 96.7% and 103.3% compared with Presto, ECMP, RLJD and OSPF, when the number of flows is 15×10^4. That is consistent with the above results that PrePass not only achieves similar LLR to RLJDM, but also satisfies the FTS constraint.

From the above simulations in Figs. 11–18, we can draw some conclusions. First, PrePass can reduce LLR by 30%, 40% and 63% compared with Presto, ECMP and OSPF for both two topologies and workloads. Second, PrePass satisfies the flow table size constraint while it only increases the link load ratio by about 5% compared with RLJDM.

6.3.3. *Performance under traffic dynamics*

The aggregate path deployment determined by the predicted traffic statistics, which is the long-term traffic statistics collection on switches. Due to the flow dynamics, the prediction error is unavoidable. Thus, it is necessary to evaluate the routing performance when the prediction error exists. There are two kinds of prediction errors, the number of flows in a macroflows and the traffic size of a macroflow. The proactive scheme ensures that one macroflow needs one and only one wildcard entry on each switch along the path, no matter how many flows the macroflow includes. Thus we only consider the error of traffic size prediction. Without loss of generality, it is assumed that the controller collects the traffic size with prediction error parameter X. Specifically, note that $f(\gamma)$ and $\bar{f}(\gamma)$ are the predicted traffic size and actual traffic size of macroflow γ respectively. We will simulate the case that the predicted traffic size of macroflow γ obeys the uniform distribution from (1-X%)$\cdot\bar{f}(\gamma)$ to $\bar{f}(\gamma)$ or from $\bar{f}(\gamma)$ to (1-X%)$\cdot\bar{f}(\gamma)$. X can be positive or negative, representing under-prediction and over-prediction respectively.

This section evaluates the impacts of traffic dynamics. According to Figs. 11 and 12, when the number of flows is 12×10^4, the number of adjustable flows is about 4.5×10^4. It means the ratio of adjustable flows to all flows is 37.5%. Since most of the flows are forwarded through aggregate paths, we choose 12×10^4 to conduct our simulations by changing the prediction error parameter X from -100 to 100. Since ECMP, Presto and RLJDM make routing decisions in flow level, not macroflow level, they will not be affected by the prediction error and remain fixed. The simulation results in terms of LLR are shown in Figs. 19 and 20. When the error parameter X is 0, the gap between RLJDM and PrePass is the smallest. When $|X|$ increases, the gap increases too. For example, the gap between PrePass and RLJDM increases from 5% to 10% when X increases from 0 to 100 by Fig. 19(a). What's more, PrePass outperforms Presto and ECMP, even when the prediction error parameter X is 100. That is because PrePass routes partial flows with switch-host granularity to highly utilize the link capacity and reduce the prediction error caused by aggregate paths. In a word, PrePass can well adapt to the flow traffic dynamics and uncertainty, and achieve

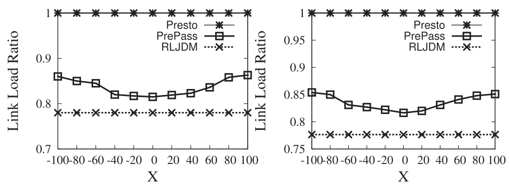


Fig. 20. Prediction error parameter X vs. LLR in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

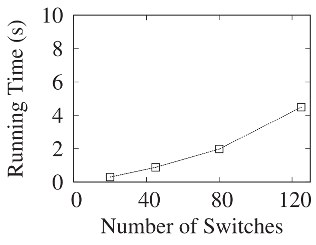


Fig. 21. Running time of PrePass under different scales of networks.

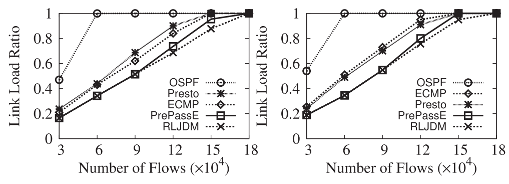


Fig. 22. No. of flows vs. link load ratio (LLR) in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

the similar LLR (less than 10%) to RLJDM even when the prediction error parameter X is 100.

6.3.4. Running time of PrePass

We evaluate the computing cost of PrePass in different scales of networks. The topologies is fat-tree as it is scalable. The results in Fig. 21 show that PrePass requires more time to solve the routing scheme when the number of switches increases. In spite of this, we stress that PrePass can solve the problem in time even when the network is very large. For instance, in our simulation with 80-switch topology, the proactive scheme will be triggered every 10min, which is much longer compared with the computing cost of PrePass (1.98 s). Therefore, we can conclude that PrePass is practical and PrePass can perform proactive routing policy efficiently in the beginning of each period.

6.3.5. Performance for PrePassE

This section evaluates the performance of the PrePassE algorithm, in which the controller is oblivious to the traffic size of each flow/macroflow. For simplicity, the integrated routing algorithm, combing PrePassE and reactive scheme, is also called PrePassE. Since we focus on the load balancing in this paper, this section mainly observes the routing performance of PrePassE.

The first set of simulations observes the link load ratio performance and the results are shown in Figs. 22 and 23. As are result, PrePassE can reduce LLR by about 25%, 29.5%, and 60% compared with Presto, ECMP and OSPF, respectively. Moreover, PrePassE can achieve almost the same LLR as RLJDM when the number of flows is less than 9×10^4. When the number of flows exceeds 9×10^4, PrePassE increases a little LLR compared with RLJDM. For example, when the number of flows is 12×10^4 and 15×10^4, PrePassE increases LLR by about 7% and

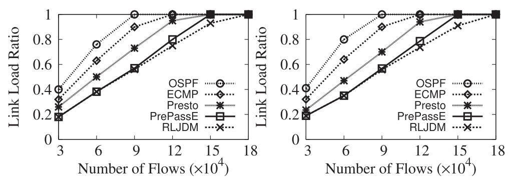


Fig. 23. No. of flows vs. link load ratio (LLR) in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

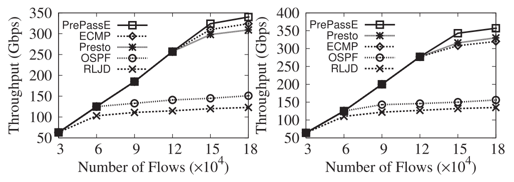


Fig. 24. Number of flows vs. throughput in topology T1. *Left plot*: enterprise workload; *right plot*: data mining workload.

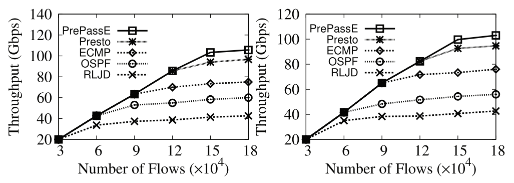


Fig. 25. Number of flows vs. throughput in topology T2. *Left plot*: enterprise workload; *right plot*: data mining workload.

10% respectively from Fig. 22(a). The second set of simulations studies the performance of network throughput. As shown in Figs. 24 and 25, PrePassE can improve network throughput compared with four benchmarks. In particular, PrePassE increases throughput by 8.0%, 35.9%, 81.9%, and 146.7% compared with Presto, ECMP, OSPF, and RLJD, when the number of flows is 15×10^4. Note that Presto needs additional virtual switches for each physical switch. From these simulation results, we can conclude that PrePassE can achieve close LLR to RLJDM, which required unlimited flow table size, and improve throughput than other benchmarks, even if the controller is oblivious to the traffic size of each flow for deployment of aggregate paths.

7. Conclusions

In this paper, we study the impact of data plane resource constraints on load balancing in SDNs. To overcome this challenge, we propose PrePass, which combines proactive routing and reactive routing with different granularities in an SDN, to satisfy data plane resource constraints. We formulate the load balancing with flow table size constraint (LB-FTS) problem as an integer linear program. A rounding-based algorithm, PrePass, with bounded approximation factors is proposed to solve the LB-FTS problem. We also propose an extended algorithm of PrePass, i.e., PrePassE, for load balancing without the traffic size knowledge of each flow. We implement the proposed algorithms on an SDN testbed for experimental studies and use simulations for large-scale evaluation. The experimental results and extensive simulation results show that PrePass (including PrePassE) can satisfy the different resource constraints on switches, and only increase the link load ratio by about 5%-10% compared with per-flow routing scheme under various scenarios. Our simulation results also show that PrePass can well handle the traffic dynamics, increasing the link load ratio by less than 10% even when the traffic size prediction produces error is very large.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Haibo Wang: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft. **Hongli Xu:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision, Funding acquisition, Project administration. **Chen Qian:** Conceptualization, Writing - review & editing, Funding acquisition. **Juncheng Ge:** Software. **Jianchun Liu:** Software. **He Huang:** Writing - review & editing, Funding acquisition.

Acknowledgement

This research is supported by the National Science Foundation of China (NSFC) under Grants 61822210, 61936015, and U1709217; and by Anhui Initiative in Quantum Information Technologies under No. AHY150300.

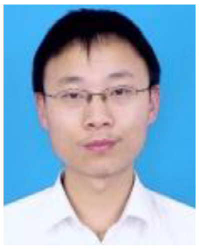
Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2020.107339.

References

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, Hedera: dynamic flow scheduling for data center networks., in: NSDI, 10, 2010, p. 19.
[2] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven wan, in: ACM SIGCOMM, 2013, pp. 15–26.
[3] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S. Banerjee, DevoFlow: scaling flow management for high-performance networks, in: ACM SIGCOMM Computer Communication Review, 2011, pp. 254–265.
[4] J. Liu, Y. Li, D. Jin, SDN-based live VM migration across datacenters, in: ACM SIGCOMM Computer Communication Review, ACM, 2014, pp. 583–584.
[5] H. Owens II, A. Durresi, Video over software-defined networking (VSDN), COMNET 92 (2015) 341–356.
[6] B. Pfaff, et al., OpenFlow switch specification v1.3.0, 2012.
[7] A. Wang, Y. Guo, F. Hao, T. Lakshman, S. Chen, Scotch: elastically scaling up SDN control-plane using vswitch based overlay, in: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, ACM, 2014, pp. 403–414.
[8] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, VL2: a scalable and flexible data center network, in: ACM SIGCOMM Computer Communication Review, 2009, pp. 51–62.
[9] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review, ACM, 2008, pp. 63–74.
[10] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, A. Vahdat, WCMP: weighted cost multipathing for improved fairness in data centers, in: Proceedings of the Ninth European Conference on Computer Systems, ACM, 2014, p. 5.
[11] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese, et al., CONGA: distributed congestion-aware load balancing for datacenters, in: ACM SIGCOMM Computer Communication Review, ACM, 2014, pp. 503–514.
[12] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, D. Maltz, Per-packet load-balanced, low-latency routing for clos-based data center networks, in: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, ACM, 2013, pp. 49–60.
[13] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, R. Fonseca, Planck: millisecond-scale monitoring and control for commodity networks, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2015) 407–418.
[14] H. Xu, H. Huang, S. Chen, G. Zhao, Scalable software-defined networking through hybrid switching, in: Proc. IEEE INFOCOM, 2017.
[15] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, A. Akella, Presto: edge-based load balancing for fast datacenter networks, ACM SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 465–478.
[16] G. Zhao, H. Xu, S. Chen, L. Huang, P. Wang, Deploying default paths by joint optimization of flow table and group table in sdns, in: Proc. IEEE ICNP, 2017.
[17] R. Cohen, L. Lewin-Eytan, J.S. Naor, D. Raz, On the effect of forwarding table size on SDN network utilization, in: INFOCOM, 2014 Proceedings IEEE, IEEE, 2014, pp. 1734–1742.
[18] S. Ghorbani, B. Godfrey, Y. Ganjali, A. Firoozshahian, Micro load balancing in data centers with drill, in: Proceedings of the 14th ACM Workshop on Hot Topics in Networks, ACM, 2015, p. 17.
[19] S. Sen, D. Shue, S. Ihm, M.J. Freedman, Scalable, optimal flow routing in datacenters via local link balancing, in: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, 2013, pp. 151–162.
[20] N. Katta, M. Hira, C. Kim, A. Sivaraman, J. Rexford, HULA: Scalable load balancing using programmable data planes, in: Proceedings of the Symposium on SDN Research, ACM, 2016, p. 10.
[21] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, T. Edsall, Let it flow: resilient asymmetric load balancing with flowlet switching., in: NSDI, 2017, pp. 407–420.
[22] G. Zhao, L. Huang, Z. Yu, H. Xu, P. Wang, On the effect of flow table size and controller capacity on SDN network throughput, in: ICC, 2017 IEEE International Conference on, 2017, pp. 1–6.
[23] S. Banerjee, K. Kannan, Tag-in-tag: efficient flow table management in SDN switches, in: Network and Service Management (CNSM), 2014 10th International Conference on, IEEE, 2014, pp. 109–117.
[24] P. Gill, N. Jain, N. Nagappan, Understanding network failures in data centers: measurement, analysis, and implications, in: ACM SIGCOMM Computer Communication Review, ACM, 2011, pp. 350–361.
[25] H. Zhang, J. Zhang, W. Bai, K. Chen, M. Chowdhury, Resilient datacenter load balancing in the wild, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017, pp. 253–266.
[26] H. Xu, H. Huang, S. Chen, G. Zhao, L. Huang, Achieving high scalability through hybrid switching in software-defined networking, IEEE/ACM Trans. Netw. 26 (1) (2018) 618–632.
[27] O. Rottenstreich, I. Keslassy, Worst-case TCAM rule expansion, in: 2010 Proceedings IEEE INFOCOM, IEEE, 2010, pp. 1–5.
[28] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, V. Braverman, One sketch to rule them all: rethinking network flow monitoring with UnivMon, in: Proceedings of the 2016 ACM SIGCOMM Conference, ACM, 2016, pp. 101–114.
[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM Comput. Commun. Rev. 38 (2) (2008) 69–74.
[30] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, S.A. Khayam, Macroflows and microflows: enabling rapid network innovation through a split SDN data plane, in: Software Defined Networking, 2012 European Workshop on, IEEE, 2012, pp. 79–84.
[31] Y. Peng, K. Chen, G. Wang, W. Bai, Z. Ma, L. Gu, HadoopWatch: a first step towards comprehensive traffic forecasting in cloud computing, in: INFOCOM, 2014 Proceedings IEEE, pp. 19–27.
[32] S. Even, A. Itai, A. Shamir, On the complexity of time table and multi-commodity flow problems, in: Foundations of Computer Science, 1975., 16th Annual Symposium on, IEEE, 1975, pp. 184–193.
[33] P. Raghavan, C.D. Tompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, Combinatorica 7 (4) (1987) 365–374.
[34] P. L. balancing with data plane resource constraints using commodity SDN switches (extended version), (https://github.com/haiporwang/super-disco/blob/haiporwang-patch-1/LB-SDN(5).pdf).
[35] M.Y.O. Network, Ospf network design solutions(2003).
[36] S. Sahni, Approximate algorithms for the 0/1 knapsack problem, J. ACM (JACM) 22 (1) (1975) 115–124.
[37] N. Dukkipati, N. McKeown, Why flow-completion time is the right metric for congestion control, ACM SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 59–62.
[38] D. Zats, T. Das, P. Mohan, D. Borthakur, R. Katz, DeTail: reducing the flow completion time tail in datacenter networks, in: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM, 2012, pp. 139–150.
[39] O. vSwitch: open virtual switch., (http://openvswitch.org/).
[40] R.S. Framework, (http://osrg.github.io/ryu/).
[41] W. Bai, L. Chen, K. Chen, H. Wu, Enabling {ECN} in multi-service multi-queue data centers, in: 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), 2016, pp. 537–549.
[42] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, H. Wang, Information-agnostic flow scheduling for commodity data centers, in: 12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15), 2015, pp. 455–468.
[43] T. N. T. from the Monash University, (http://www.ecse.monash.edu.au/twiki/bin/view/InFocus/LargePacket-switchingNetworkTopologies).

Haibo Wang received B.E. degree in nuclear science in 2016 and Masters degree in computer science 2019, both from the University of Science and Technology of China. He is currently a Ph.D. student in Computer and Information Science and Engineering, University of Florida. His main research interest is Internet traffic measurement, software defined networks, and optical circuit scheduling. He is a student member of IEEE.



Hongli Xu is a professor in the School of Computer Science and Technology at the University of Science and Technology of China (USTC). He received his B.S. degree in Computer Science from USTC in 2002. He received Ph.D degree in Computer Software and Theory from USTC in 2007. 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. He has published more than 100 papers in famous journals and conferences, including ToN, JSAC, TMC, TPDS, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interest is software defined networks, edge computing and Internet ofThing. He is a member of the IEEE.

Chen Qian (M08) received the B.S. degree from Nanjing University in 2006, the M.Phil. degree from The Hong Kong University of Science and Technology in 2008, and the Ph.D. degree from The University of Texas at Austin in 2013, all in computer science. He is currently an Assistant Professor with the Department of Computer Engineering, University of California at Santa Cruz. His research interests include computer networking, network security, and Internet of Things. He has authored over 60 research papers in highly competitive conferences and journals. He is a member of the ACM.

Juncheng Ge is a master student in the School of Software Engineering at the University of Science and Technology of China. His research interests are in the areas of software defined networking.

Jianchun Liu is currently a Ph.D. student in the School of Computer Science and Technology at the University of Science and Technology of China. His research interests are in the areas of software defined networking, network function virtualiztion and edge computing.

He Huang is a professor in the School of Computer Science and Technology at Soochow University, Soochow, China. He received the Ph.D. degree in Department of Computer Science and Technology from University of Science and Technology of China in 2011. His current research interests include spectrum auction, privacy preserving in auction, wireless sensor networks, and algorithmic game theory. He is a member of IEEE computer society, and a member of ACM.