POFM: Profit-Oriented Flow Management in SDN-Based Data Center Networks

You-Chiun Wang[1,2] and Jhong-Ting Liang[1]
[1]Department of Computer Science and Engineering; [2]Information Security Research Center
National Sun Yat-sen University, Kaohsiung, Taiwan
Email: ycwang@cse.nsysu.edu.tw; m103040119@nsysu.edu.tw

***Abstract*—In a *data center network (DCN)*, there are numerous hosts connected by well-organized switches, usually involving a large amount of data transmission to offer cloud services. How to manage flows in a DCN by deciding packet routes is critical. This paper applies the *software-defined networking (SDN)* technique to flow management in fat-tree DCNs. Unlike previous studies, we consider that flows have different values and associate flows with profits to reflect their values. The practical profit acquired from each flow will depend on the proportion of successfully sent packets. The paper proposes a *profit-oriented flow management (POFM)* scheme to maximize the profit obtained while reducing packet loss in flows. In POFM, once a link becomes congested, some flows on that link may be rerouted to other paths. When no alternative path can be found, POFM adjusts the data rates of flows on the congested link based on multiple factors, including their demanded rates, profits, and transmission durations. With simulations, we show that POFM can effectively raise the overall profit and decrease the packet loss rate in the DCN.**

***Keywords*—congestion, data center network (DCN), fat tree, flow management, software-defined networking (SDN).**

I. Introduction

To deliver services in cloud computing and big data, *data center networks (DCNs)* are deployed at large. In one DCN, many hosts are connected via switches in a well-structured topology like a fat tree to offer scalability [1]. As the amount of traffic is huge, how to route flows between hosts (referred to as *flow management*) is important. However, conventional switches relay packets using a given routing protocol. In cases of congestion, network managers often have to modify packet routing by manually configuring some switches, which incurs a high cost and degrades performance.

Software-defined networking (SDN) is a technique of network management, placing control over switches via a controller. The controller builds secure connections with switches and communicates with them using the OpenFlow protocol [2]. On one hand, the controller obtains the network status by inquiring about the number and types of packets handled by switches. On the other hand, it can issue commands to switches to manipulate transmissions of packets. This allows users to direct switches by installing their programs on the controller, which facilitates network management.

Applying SDN to flow management in DCNs has gained attention. Many methods guide flows on congested links to others for load balance. They consider that flows possess the same value and attempt to reduce packet loss. In effect, flows have different values. For example, firm real-time flows (e.g., video conferences) and important backup flows (e.g., virtual machine snapshots) should have high values. On the contrary, flows of web connections or video streaming whose quality is reducible can be given low values. Note that value is different from priority. Low-priority flows may be preempted by high-priority flows when they vie for the bandwidth of a congested link. In contrast, low-value and high-value flows are capable of sharing link bandwidth to avoid starving low-value flows.

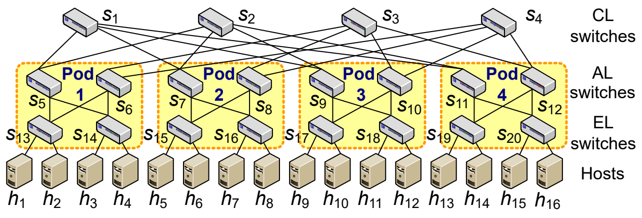
Consequently, we introduce the concept of profit. Every flow is associated with a profit based on its value. The actual profit that we get from a flow depends on the proportion of successfully sent packets. Then, this paper proposes a *profit-oriented flow management (POFM)* scheme whose objectives are to maximize the obtained profit while reducing packet loss in flows. When a link becomes congested, POFM searches for *alternative paths (APs)* to reroute some flows on that link. If there is no available AP, POFM adjusts the data rate of each flow on the congested link based on its demanded rate, profit, and *transmission duration (TD)*. Simulation results reveal that the POFM scheme can efficiently increase the overall profit gained and decrease the *packet loss rate (PLR)* of flows.

The residual of this paper is outlined as follows: Section II discusses related work, and Section III describes the system model. The POFM scheme is detailed in Section IV, followed by its performance evaluation in Section V. Then, Section VI contains concluding remarks.

II. Related Work

Various issues for DCN management using SDN are discussed. The study [3] proposes a resource allocation method for the DCN's physical layer. Liu et al. [4] deploy multiple controllers to provide a distributed control structure for DCN congestion monitoring. The work [5] assigns clients' requests to servers in a DCN to balance their loads and save response time for requests. Both [6] and [7] identify flows carrying large quantities of data. Qin et al. [8] differentiate between long and short flows on the host side. The above studies have different objectives than ours.

Many flow management approaches are designed for SDN-based DCNs. Kumar et al. [9] balance link loads using round-robin and Dijkstra-based strategies. The work [10] determines if some links are congested depending on the load variance and redirects traffic to light-load links. In [11], once a path experiences congestion, it is replaced by another path whose total link cost is the smallest. The study [12] splits flows by forwarding packets via different paths to alleviate congestion. Fan et al. [13] employ a roulette wheel to select new paths for flows according to the local load. The study [14] finds paths for flows using depth-first search, where the weight of each link is computed by its available bandwidth. The work [15] modifies the *equal-cost multi-path (ECMP)* protocol by



s_1
s_2
s_3
s_4
CL switches
AL switches
EL switches
Hosts
Pod 1
Pod 2
Pod 3
Pod 4
s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12}
s_{13} s_{14} s_{15} s_{16} s_{17} s_{18} s_{19} s_{20}
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9 h_{10} h_{11} h_{12} h_{13} h_{14} h_{15} h_{16}

Fig. 1. 4-ary fat-tree DCN.

TABLE I
SUMMARY OF ACRONYMS AND NOTATIONS.

(a) acronyms

acronym	full name
AL/CL/EL	aggregation/core/edge layer
AP	alternative path
BML	bandwidth-based multipath load-balancing
DCN	data center network
ECMP	equal-cost multi-path (P-ECMP: predictive ECMP)
PLR	packet loss rate
POFM	profit-oriented flow management
SDN	software-defined networking
TD	transmission duration

(b) notations

notation	definition
$\hat{\mathcal{F}}$, $\hat{\mathcal{F}}_k$	sets of flows in the DCN and on link l_k ($\hat{\mathcal{F}}_k \subseteq \hat{\mathcal{F}}$)
$\hat{\Gamma}_i$	set of shortest paths for flow f_i
N_i^G, N_i^L	numbers of flow f_i's generating and lost packets
b_i, ζ_i, τ_i	flow f_i's bandwidth requirement, profit, and TD
λ_i^D, λ_i^R	demanded and deducted rates of flow f_i
c_k, u_k	capacity and bandwidth usage of link l_k

making forwarding decisions based on predicted congestion outlooks. In [16], flows are divided into three ranks and have different priorities for choosing links to send packets.

Compared to the prior work, our work considers that flows have different values and thus proposes the POFM scheme to efficiently increase the profit obtained and reduce packet loss in flows. This distinguishes our work from the prior work.

III. System Model

We are given one DCN whose topology is a K-ary fat tree. Each switch has at most K ports. All switches are organized into a three-layer architecture: *core layer (CL)*, *aggregation layer (AL)*, and *edge layer (EL)*. Fig. 1 illustrates an example by setting $K = 4$. EL and AL switches are grouped into K *pods*, where each pod consists of $K/2$ EL switches and $K/2$ AL switches. Hence, each EL switch can link with $K/2$ AL switches and $K/2$ hosts. Moreover, every CL switch connects with an AL switch in each pod. There will be no more than $K^2/4$ CL switches. A fat tree is capable of interconnecting $K^3/4$ hosts using $5K^2/4$ switches. This topology possesses two advantages [17]: 1) adding switches and hosts becomes easy, and 2) there can be multiple shortest paths between any two hosts in the DCN.

To utilize SDN for the DCN's management, one controller coordinates it. Switches can support OpenFlow and maintain flow tables to store flow entries issued by the controller [18]. Then, whenever a packet arrives, the switch checks if it fulfills the conditions specified in the match-fields of a flow entry. If so, the switch processes the packet following that entry's instruction. In this way, the controller can dynamically change packet routing based on the network status. Moreover,

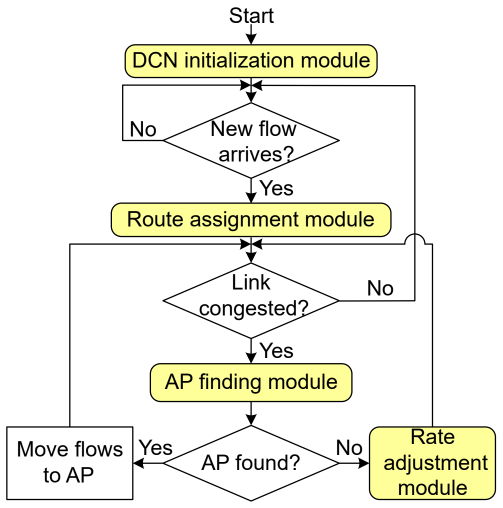


Fig. 2. Flowchart of the POFM scheme.

each switch has a meter table used to determine the data rates of flows. The controller installs meter entries in this table to set the traffic limit of a flow passing that switch.

Let $\hat{\mathcal{F}}$ be the set of flows in the DCN. Each flow $f_i \in \hat{\mathcal{F}}$ generates N_i^G packets, and N_i^L packets are lost (e.g., dropped by a switch because of link congestion). Moreover, f_i is given a profit ζ_i to reflect its value, which can be set according to application needs. Then, our problem is expressed as follows:

$$\text{maximize} \quad \sum\nolimits_{f_i \in \hat{\mathcal{F}}} \zeta_i \times \left(1 - N_i^L / N_i^G\right), \tag{1}$$

$$\text{minimize} \quad \sum\nolimits_{f_i \in \hat{\mathcal{F}}} N_i^L \Big/ \sum\nolimits_{f_i \in \hat{\mathcal{F}}} N_i^G, \tag{2}$$

subject to

$$0 \le N_i^L \le N_i^G \qquad \forall f_i \in \hat{\mathcal{F}}, \tag{3}$$

$$\zeta_{\min} \le \zeta_i \le \zeta_{\max} \qquad \forall f_i \in \hat{\mathcal{F}}. \tag{4}$$

The objective function in Eq. (1) is to maximize the overall profit, where the actual profit that we get from f_i is proportional to the proportion of its successfully sent packets. The objective function in Eq. (2) is to minimize the PLR. Note that Eq. (2) is also used to avoid a greedy solution where high-profit flows are always served first, making low-profit flows starve. Regarding constraints, Eq. (3) indicates that the lost packets in each flow shall be no more than the total packets that it produces. Eq. (4) gives upper and lower bounds on a flow's profit. Table I lists acronyms and notations adopted in this paper.

IV. The Proposed POFM Scheme

Fig. 2 gives POFM's flowchart. The controller collects link states from switches through the *DCN initialization module*. Then, when a new flow arrives, the *route assignment module* is invoked to find a suitable route for the flow. If some links become congested, we search for APs to reroute some flows on congested links. This is done using the *AP finding module*. However, if no AP can be found, the *rate adjustment module* is adopted to decide the data rate for each flow on a congested link. Below, we detail each module and then have a discussion on the POFM scheme.

A. DCN Initialization Module

The controller can discover links in the DCN using the *link layer discovery protocol (LLDP)* [19]. Specifically, it employs a Packet_Out message to transmit an LLDP request to each switch. Then, the switch places the LLDP reply (containing link information) in a Packet_In message and sends it back to the controller. This way, the controller can know the existence of each link and thus grasp the DCN's topology.

The controller maintains a *link state table* to keep track of the status of each link. For a link $l_k = [s_i, s_j]$ between two switches s_i and s_j, there is an entry $\langle [s_i, s_j], (p_x, p_y), c_k, u_k \rangle$ in the link state table, where s_i uses its port p_x to connect with s_j's port p_y. Moreover, c_k is the capacity (i.e., the maximum bandwidth) of l_k, and u_k is the amount of bandwidth usage. Both c_k and u_k are measured in Mbps. When l_k is the link between a switch s_i and a host h_j, the format of its entry is $\langle [s_i, h_j], (p_x, \text{NIL}), c_k, u_k \rangle$, where NIL denotes null.

To calculate u_k, the controller can send a *PortStatsRequest* message to s_i to query the number of data bytes transmitted and received through its port p_x. Afterward, s_i replies with a *PortDescStatsReply* message that includes the answer. Let $B^{\texttt{TX}}_{\texttt{now}}$ and $B^{\texttt{TX}}_{\texttt{pre}}$ be the numbers of data bytes sent mentioned in the current answer and the answer in the last querying period, respectively. Moreover, let $B^{\texttt{RX}}_{\texttt{now}}$ and $B^{\texttt{RX}}_{\texttt{pre}}$ denote the numbers of data bytes received indicated in the current answer and the answer in the last querying period, respectively. Then, u_k can be derived as follows:

$$u_k = \frac{(B^{\texttt{TX}}_{\texttt{now}} - B^{\texttt{TX}}_{\texttt{pre}} + B^{\texttt{RX}}_{\texttt{now}} - B^{\texttt{RX}}_{\texttt{pre}}) \times 8}{\Delta_Q \times 10^6}, \tag{5}$$

where Δ_Q is the length of a querying period (in seconds).

B. Route Assignment Module

Using the depth-first search algorithm, the controller finds all the shortest paths between any two hosts in the DCN and stores these paths in a *candidate path table*. Let $\hat{\Gamma}_i$ be the set of all shortest paths for a flow $f_i \in \hat{\mathcal{F}}$. Let us take Fig. 1 as an example, where f_i is sent from host h_1 to host h_8. Then, we have $\hat{\Gamma}_i = \{h_1 \Rightarrow s_{13} \Rightarrow s_5 \Rightarrow s_1 \Rightarrow s_7 \Rightarrow s_{16} \Rightarrow h_8, h_1 \Rightarrow s_{13} \Rightarrow s_5 \Rightarrow s_2 \Rightarrow s_7 \Rightarrow s_{16} \Rightarrow h_8, h_1 \Rightarrow s_{13} \Rightarrow s_6 \Rightarrow s_3 \Rightarrow s_8 \Rightarrow s_{16} \Rightarrow h_8, h_1 \Rightarrow s_{13} \Rightarrow s_6 \Rightarrow s_4 \Rightarrow s_8 \Rightarrow s_{16} \Rightarrow h_8\}$.

Let b_i denote the amount of bandwidth required by flow f_i. Algorithm 1 presents the pseudocode of the route assignment module. We adopt a set $\hat{\Gamma}'_i$ to store each path γ_j in $\hat{\Gamma}_i$ such that every link l_k in γ_j can satisfy two conditions: 1) l_k has enough residual bandwidth to serve f_i (i.e., $c_k - u_k \geq b_i$) and 2) l_k's bandwidth consumption ratio is below a threshold δ (i.e., $u_k/c_k \leq \delta$), where $0 < \delta < 1$. Here, condition 2 is to reduce the chance of congestion occurring. The code is given in lines 1–6. Then, if $\hat{\Gamma}'_i$ is not empty, we adopt the best-fit strategy by choosing the path whose busiest link (i.e., with the minimum value of $c_k - u_k$) possesses the least residual bandwidth, as shown in line 8.

However, if no path in $\hat{\Gamma}_i$ meets the above two conditions, line 9 picks a path whose busiest link has the most residual bandwidth. Note that the path γ_j selected by line 9 may not have enough bandwidth to serve f_i, causing congestion. In this case, we can use the AP finding module in Section IV-C to change the routes of some flows on γ_j's congested links.

Algorithm 1: Route Assignment Module

```
1  $\hat{\Gamma}'_i \leftarrow \hat{\Gamma}_i$;
2  foreach $\gamma_j \in \hat{\Gamma}'_i$ do
3      foreach $l_k \in \gamma_j$ do
4          if $c_k - u_k < b_i$ or $u_k/c_k > \delta$ then
5              $\hat{\Gamma}'_i \leftarrow \hat{\Gamma}'_i \setminus \{\gamma_j\}$;
6              break;
7  if $\hat{\Gamma}'_i \neq \emptyset$ then
8      return $\arg\min_{\gamma_j \in \hat{\Gamma}'_i} \min_{\forall l_k \in \gamma_j} c_k - u_k$;
9  return $\arg\max_{\gamma_j \in \hat{\Gamma}_i} \min_{\forall l_k \in \gamma_j} c_k - u_k$;
```

Algorithm 2: AP Finding Module

```
1  pick a flow $f_i \in \hat{\mathcal{F}}_k$ with the maximum demand $b_i$;
2  $\hat{\Gamma}'_i \leftarrow \hat{\Gamma}_i \setminus \hat{\Gamma}_{i,k}$;
3  foreach $\gamma_j \in \hat{\Gamma}'_i$ do
4      foreach $l_x \in \gamma_j$ do
5          if $c_x - u_x < b_i$ then
6              $\hat{\Gamma}'_i \leftarrow \hat{\Gamma}'_i \setminus \{\gamma_j\}$;
7              break;
8  if $\hat{\Gamma}'_i \neq \emptyset$ then
9      $\gamma_j \leftarrow \arg\min_{\gamma_j \in \hat{\Gamma}'_i} \min_{\forall l_x \in \gamma_j} c_x - u_x$;
10     return $f_i$ and $\gamma_j$;
11 return null;
```

C. AP Finding Module

Let $\hat{\mathcal{F}}_k$ be the set of flows on a link l_k. If the sum demand for bandwidth of all flows in $\hat{\mathcal{F}}_k$ exceeds l_k's capacity (i.e., $\sum_{f_i \in \hat{\mathcal{F}}_k} b_i > c_k$), l_k is congested. In this case, the controller finds APs to reroute some flows in $\hat{\mathcal{F}}_k$ to mitigate congestion.

Algorithm 2 displays the pseudocode for the AP finding module. Specifically, line 1 picks the flow f_i (on link l_k) with the maximum demand b_i to handle, as it is the main cause of link congestion. Given set $\hat{\Gamma}_i$ of all shortest paths for f_i, we exclude those paths from $\hat{\Gamma}_i$ that contain the congested link l_k (as denoted by $\hat{\Gamma}_{i,k}$, which can be found in the candidate path table). Then, the remaining paths are stored in a set $\hat{\Gamma}'_i$, as shown in line 2. The code in lines 3–7 removes the paths from $\hat{\Gamma}'_i$ without enough residual bandwidth to serve f_i (since they will be inevitably congested after accommodating f_i).

The remaining paths in $\hat{\Gamma}'_i$ will be candidate APs. Among them, we select an AP based on the best-fit strategy (i.e., its busiest link has the least residual bandwidth), as shown in line 9. Then, line 10 returns both f_i and γ_j. Hence, the controller can move flow f_i to AP γ_j for mitigating congestion on link l_k. If $\hat{\Gamma}'_i = \emptyset$, it implies that there is no AP, so this module returns null in line 11. As referring to the flowchart in Fig. 2, the rate adjustment module in Section IV-D will then be used to alleviate congestion.

D. Rate Adjustment Module

This module adjusts the data rates of flows on a congested link l_k if no APs can be used to reroute these flows. Given a

TABLE II
EXAMPLE OF FINDING DEDUCTED RATES.

flow	demanded rate	profit	current TD
f_1	$\lambda_1^D = 10\,\text{Mbps}$	$\zeta_1 = 70$	$\tau_1 = 12\,\text{s}$
f_2	$\lambda_2^D = 40\,\text{Mbps}$	$\zeta_2 = 20$	$\tau_2 = 6\,\text{s}$
f_3	$\lambda_3^D = 80\,\text{Mbps}$	$\zeta_3 = 30$	$\tau_3 = 1\,\text{s}$

set $\hat{\mathcal{F}}_k$ of flows on l_k, a naive solution is to favor high-profit flows by keeping their data rates and reducing the data rates of low-profit flows. Nevertheless, this solution may lead to a high PLR. In fact, it may not maximize the profit obtained. Let us consider an example, where $\hat{\mathcal{F}}_k$ contains two flows f_1 and f_2 whose demanded rates are 80 Mbps and 40 Mbps and profits are 12 and 8, respectively. Suppose that l_k's capacity is 100 Mbps. The naive solution deducts all the extra 20 Mbps from f_2. In other words, the data rates of f_1 and f_2 become 80 Mbps and 20 Mbps, respectively. In this case, f_2's PLR is $20/40 = 50\%$. Besides, the profit gained will be $12 \times (1 - 0) + 8 \times (1 - 0.5) = 16$. On the other hand, if we deduct the extra 20 Mbps from f_1, its PLR is $20/80 = 25\%$. The profit gained is $12 \times (1 - 0.25) + 8 \times (1 - 0) = 17$. As can be seen, the PLR in the naive solution is higher, but the profit gained is not the maximum. Hence, we need to design a more delicate method to determine the data rate of each flow.

Let λ_i^D and λ_i^R denote the data rates required by and to be reduced from a flow $f_i \in \hat{\mathcal{F}}_k$, referred to as the *demanded rate* and *deducted rate*, respectively. The modified data rate of f_i is $\lambda_i^D - \lambda_i^R$. Moreover, τ_i is the current TD of f_i (i.e., the amount of time that f_i's packets have been sent before congestion). Suppose that link l_k will be congested for an amount Δ_C of time. Then, f_i's average PLR can be estimated as follows:

$$\vartheta_i = \frac{\Delta_C \times \lambda_i^R/\lambda_i^D}{\tau_i + \Delta_C}. \tag{6}$$

According to Eq. (1), our objective is to maximize the profit gained from flows in $\hat{\mathcal{F}}_k$, that is,

$$\text{maximize} \quad \sum\nolimits_{f_i \in \hat{\mathcal{F}}_k} \zeta_i \times (1 - \vartheta_i), \tag{7}$$

subject to

$$0 \leq \lambda_i^R \leq \lambda_i^D \qquad \forall f_i \in \hat{\mathcal{F}}_k, \tag{8}$$

$$\sum\nolimits_{f_i \in \hat{\mathcal{F}}_k} \lambda_i^R = \sum\nolimits_{f_i \in \hat{\mathcal{F}}_k} \lambda_i^D - c_k. \tag{9}$$

Here, Eq. (8) means that the deducted rate cannot exceed the demanded rate of each flow. Eq. (9) points out that the total deducted rate of flows in $\hat{\mathcal{F}}_k$ is equal to the overall demanded rate of these flows minus link capacity (i.e., c_k). To compute λ_i^R for each flow $f_i \in \hat{\mathcal{F}}_k$, we can use the sequential least squares programming [20].

Table II gives an example, where $c_k = 100\,\text{Mbps}$. Suppose that link l_k will be congested for 1 s (i.e., $\Delta_C = 1\,\text{s}$). Then, the objective is

$$\text{maximize} \quad 70 \times \left(1 - \frac{1 \times \frac{\lambda_1^R}{10}}{12+1}\right) + 20 \times \left(1 - \frac{1 \times \frac{\lambda_2^R}{40}}{6+1}\right) + 30 \times \left(1 - \frac{1 \times \frac{\lambda_3^R}{80}}{1+1}\right),$$

TABLE III
TOOLS USED TO CONSTRUCT THE SIMULATION.

item	tool	version
operating system	Ubuntu	16.04
network simulator	Mininet	2.3.1b4
controller	Ryu	3.27
switch	Open vSwitch	3.1.2
southbound protocol	OpenFlow	1.6
flow generation	iPerf	2.1.9

subject to

$$0 \leq \lambda_1^R \leq 10, \quad 0 \leq \lambda_2^R \leq 40, \quad 0 \leq \lambda_3^R \leq 80,$$
$$\lambda_1^R + \lambda_2^R + \lambda_3^R = 10 + 40 + 80 - 100 = 30.$$

The answer will be $\lambda_1^R = 0\,\text{Mbps}$, $\lambda_2^R = 30\,\text{Mbps}$, and $\lambda_3^R = 0\,\text{Mbps}$. Hence, the data rates of f_1, f_2, and f_3 are adjusted to 10 Mbps, $40 - 30 = 10$ Mbps, and 80 Mbps, respectively. The profit gained from these three flows is 117.85.

In practice, we cannot know the correct value of Δ_C (i.e., link l_k's actual congestion time) as flows have different (and unknown) termination times. For implementation, we can set Δ_C to one constant, small value (e.g., $\Delta_C = 1\,\text{s}$). After Δ_C amount of time, if l_k is still congested, we can recalculate the deducted rates for flows in $\hat{\mathcal{F}}_k$.

E. Discussion

We discuss the rationale of our POFM scheme. In addition to the DCN initialization module (used to gain link states), POFM has three modules to manage flows. Specifically, the route assignment module picks an adequate (shortest) path for each new flow. If the selected path contains busy links, the AP finding module chooses APs with enough bandwidth to share some flows on congested links. Both modules adopt the best-fit strategy to let some links reserve as much bandwidth as possible. Doing so can help increase the chance of finding suitable paths for large-sized flows (e.g., elephant flows [21]) and reduce their packet drop accordingly.

When no AP can be found, it means that each shortest path for a flow has some congested links. To mitigate congestion, the most feasible solution is to limit the data rates of flows on a congested link. Instead of prioritizing high-profit flows, the rate adjustment module takes account of the demanded rate, profit, and TD of each flow. This way, POEM will not only avoid starving low-profit flows, but also achieve the objectives in Eqs. (1) and (2) as much as possible.

V. PERFORMANCE EVALUATION

With Mininet [22], we build a 4-ary fat-tree DCN for performance evaluation, whose topology is shown in Fig. 1. To apply SDN to the simulated DCN, the controller and switches are implemented by adopting the Ryu SDN framework [23] and the Linux Open vSwitch module [24]. Moreover, OpenFlow is the southbound protocol for the controller to interact with switches. We also employ the iPerf tool [25] to produce UDP (user datagram protocol) flows. The simulation is built on an Ubuntu operating system. Table III lists the simulation tools and their versions.

Seven flows are generated in the simulation. In particular, we have $\hat{\mathcal{F}} = \{f_1 : (h_1, h_5), f_2 : (h_3, h_7), f_3 : (h_{10}, h_6), f_4 : (h_4, h_8), f_5 : (h_{12}, h_8), f_6 : (h_{16}, h_8), f_7 : (h_{13}, h_8)\}$. Here,

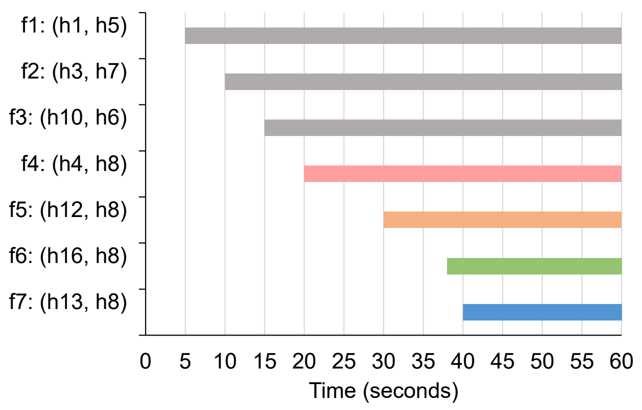


Fig. 3. Flow generation in the simulation.

the two hosts in each pair of parentheses represent the source and destination of a flow. Fig. 3 shows the overall TD of each flow, including its start time and end time. The controller does not know this information in advance. In the DCN, each link has a capacity of 100 Mbps.

Let us consider six scenarios, where flows possess different demanded rates (i.e., λ_i^D) and profits (i.e., ζ_i), as follows:

A. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (10\,\text{Mbps}, 20)$, $(\lambda_5^D, \zeta_5) = (20\,\text{Mbps}, 10)$, $(\lambda_6^D, \zeta_6) = (40\,\text{Mbps}, 40)$, $(\lambda_7^D, \zeta_7) = (60\,\text{Mbps}, 80)$.

B. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (10\,\text{Mbps}, 80)$, $(\lambda_5^D, \zeta_5) = (20\,\text{Mbps}, 40)$, $(\lambda_6^D, \zeta_6) = (40\,\text{Mbps}, 20)$, $(\lambda_7^D, \zeta_7) = (60\,\text{Mbps}, 10)$.

C. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (10\,\text{Mbps}, 10)$, $(\lambda_5^D, \zeta_5) = (20\,\text{Mbps}, 10)$, $(\lambda_6^D, \zeta_6) = (40\,\text{Mbps}, 10)$, $(\lambda_7^D, \zeta_7) = (60\,\text{Mbps}, 10)$.

D. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (60\,\text{Mbps}, 20)$, $(\lambda_5^D, \zeta_5) = (40\,\text{Mbps}, 10)$, $(\lambda_6^D, \zeta_6) = (20\,\text{Mbps}, 40)$, $(\lambda_7^D, \zeta_7) = (10\,\text{Mbps}, 80)$.

E. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (60\,\text{Mbps}, 80)$, $(\lambda_5^D, \zeta_5) = (40\,\text{Mbps}, 40)$, $(\lambda_6^D, \zeta_6) = (20\,\text{Mbps}, 20)$, $(\lambda_7^D, \zeta_7) = (10\,\text{Mbps}, 10)$.

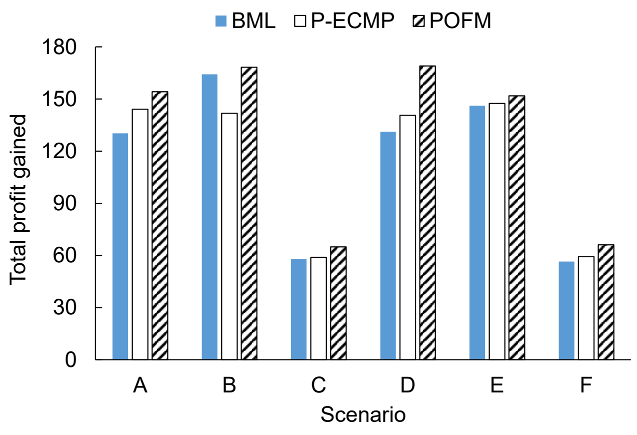
F. $(\lambda_1^D, \zeta_1) = (100\,\text{Mbps}, 10)$, $(\lambda_2^D, \zeta_2) = (100\,\text{Mbps}, 10)$, $(\lambda_3^D, \zeta_3) = (100\,\text{Mbps}, 10)$, $(\lambda_4^D, \zeta_4) = (60\,\text{Mbps}, 10)$, $(\lambda_5^D, \zeta_5) = (40\,\text{Mbps}, 10)$, $(\lambda_6^D, \zeta_6) = (20\,\text{Mbps}, 10)$, $(\lambda_7^D, \zeta_7) = (10\,\text{Mbps}, 10)$.

In all scenarios, the demand rates of flows f_1, f_2, and f_3 are set to the capacity of a link (i.e., 100 Mbps). The purpose of these three flows is to exhaust the bandwidth of three CL switches, s_1, s_2, and s_3. Doing so makes other flows vie for the bandwidth of CL switch s_4. In this way, we can observe how different methods cope with link congestion. Flows f_1, f_2, and f_3 can be viewed as *background flows*, so their profits are all set to 10 (i.e., the minimum profit in the simulation) to reduce their effect on the profit gained. Except for f_1, f_2, and f_3, each flow has different combinations of demanded rate and profit in every scenario.

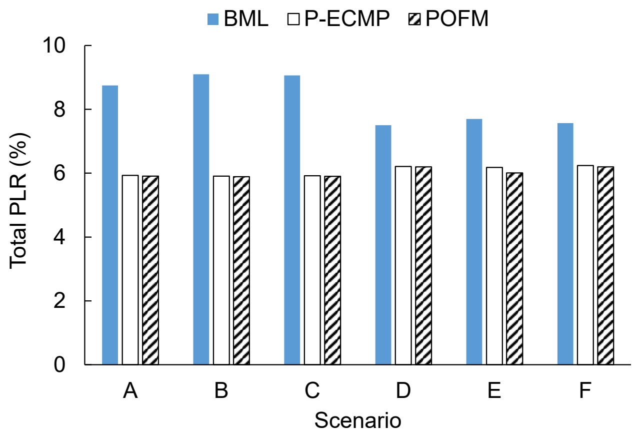
According to the flow generation in Fig. 3, the total TDs of flows f_1, f_2, f_3, f_4, f_5, f_6, and f_7 are 55 s, 50 s, 45 s, 40 s, 30 s, 22 s, and 20 s, respectively. The amount of congestion time for each flow is 20 s (in particular, from the 40th second to the 60th second).

We compare our proposed POFM scheme with two methods selected in Section II. Specifically, the *bandwidth-based multipath load-balancing (BML)* method [14] employs depth-first search to find a path for each flow. With the ECMP protocol, the *predictive ECMP (P-ECMP)* method [15] makes forwarding choices based on anticipated congestion. In POFM, we set δ to 0.5 (i.e., the threshold on bandwidth consumption ratio in Algorithm 1).

Fig. 4(a) presents the total profits obtained in different scenarios when using the BML, P-ECMP, and POFM methods. For scenarios C and F, since the profit of each flow is set to the minimum profit (i.e., 10), all methods have significantly lower profits obtained compared to other scenarios. P-ECMP generally outperforms BML in most scenarios. Scenario B is the only exception. The reason is that P-ECMP can decrease the PLRs of most flows compared to BML (the evidence will be presented later). However, P-ECMP makes high-profit flow f_4 drop more packets than BML does in scenario B. That is why P-ECMP has a lower profit than BML in that scenario. On the other hand, by taking account of the objective function in Eq. (1), POEM always has the highest profit acquired in every scenario. On average, our POEM scheme can improve 12.81% and 11.86% of profit obtained as compared with the



(a) profit gained



(b) PLR

Fig. 4. Comparison of total profit gained and PLR for each method.

BML and P-ECMP methods, respectively.

Fig. 4(b) displays the total PLR of flows in each scenario. As mentioned earlier, three background flows, f_1, f_2, and f_3, occupy the bandwidth of three CL switches, s_1, s_2, and s_3, respectively. Other flows inevitably have to compete for the bandwidth of CL switch s_4. Hence, it is difficult to find APs to reroute flows on congested links. This makes BML have the highest PLR. Then, P-ECMP conducts forwarding decisions based on predicted congestion outlooks, so it can reduce the total PLR compared to BML. Thanks to the rate adjustment module in Section IV-D, POEM will adaptively adjust the data rate of each flow on a congested link. Thus, POEM performs similarly to P-ECMP. Our POEM scheme reduces 27.31% and 0.77% of total PLR as compared to the BML and P-ECMP methods, respectively.

According to the experimental results in Fig. 4, we verify that our proposed POEM scheme can efficiently increase the overall profit while reducing PLRs of flows in a fat-tree DCN, as compared with BML and P-ECMP.

VI. Conclusion

This paper proposes the POFM scheme to manage flows in an SDN-based, fat-tree DCN. For every flow, POFM finds a suitable path and avoids overloading links. If a link becomes congested, some of its flows would be rerouted to other APs by using the best-fit strategy to mitigate congestion. When no APs can be found, POFM dynamically adjusts the data rates of flows on the congested link according to their demanded rates, profits, and TDs. Using Mininet simulations, we show that the POFM scheme not only improves the profit obtained, but also reduces the total PLR, compared with both BML and P-ECMP methods.

Regarding future work, we will take account of transmission fairness among flows, for example, achieving the Pareto optimality [26]. Furthermore, it deserves further investigation on flow management in a multi-domain SDN-based network, where each domain (or subnetwork) is directed by a controller and there are links between domains [27]. This requires the collaboration between controllers to mitigate congestion and improve throughput.

Acknowledgment

This work was supported by National Science and Technology Council, Taiwan under Grant 111-2221-E-110-023-MY2 and also the Information Security Research Center at National Sun Yat-sen University, Taiwan.

References

[1] M. Zhao, Z. Han, and X. Du, "A survey of data center network topology structure," in *International Conference on Advanced Communication Technology*, 2023, pp. 303–309.

[2] ONF, "OpenFlow specifications," https://opennetworking.org/software-defined-standards/specifications/.

[3] M. Yang, H. Rastegarfar, and I. B. Djordjevic, "Physical-layer adaptive resource allocation in software-defined data center networks," *Journal of Optical Communications and Networking*, vol. 10, no. 12, pp. 1015–1026, 2018.

[4] Y. Liu, H. Gu, Z. Zhou, and N. Wang, "RSLB: Robust and scalable load balancing in software-defined data center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4706–4720, 2022.

[5] Y. C. Wang and J. F. Zhang, "ERA: Efficient request assignment for servers in data center networks with SDN," in *IEEE International Conference on Smart Communities: Improving Quality of Life Using AI, Robotics and IoT*, 2023, pp. 24–29.

[6] F. Tang, H. Zhang, L. T. Yang, and L. Chen, "Elephant flow detection and load-balanced routing with efficient sampling and classification," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1022–1036, 2021.

[7] Y. Liu, H. Gu, and N. Wang, "HPSTOS: High-performance and scalable traffic optimization strategy for mixed flows in data center networks," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2649–2663, 2022.

[8] L. Qin, W. Wei, and X. Diao, "FlowDecider: End-host driven proactive load balancing for data center networks," in *IEEE International Conference on Communication Technology*, 2021, pp. 931–936.

[9] V. Kumar, S. Jangir, and D. G. Patanvariya, "Traffic load balancing in SDN using round-robin and Dijkstra based methodology," in *International Conference for Advancement in Technology*, 2022, pp. 1–4.

[10] Y. L. Lan, K. Wang, and Y. H. Hsu, "Dynamic load-balanced path optimization in SDN-based data center networks," in *International Symposium on Communication Systems, Networks and Digital Signal Processing*, 2016, pp. 1–6.

[11] U. Zakia and H. B. Yedder, "Dynamic load balancing in SDN-based data center networks," in *IEEE Annual Information Technology, Electronics and Mobile Communication Conference*, 2017, pp. 242–247.

[12] Y. C. Wang and S. Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1422–1434, 2018.

[13] F. Fan, H. Meng, B. Hu, K. L. Yeung, and Z. Zhao, "Roulette wheel balancing algorithm with dynamic flowlet switching for multipath datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 834–847, 2021.

[14] F. Chahlaoui, H. Dahmouni, and H. E. Alami, "Multipath-routing based load-balancing in SDN networks," in *Conference on Cloud and Internet of Things*, 2022, pp. 180–185.

[15] E. Nepolo and G. A. L. Zodi, "A predictive ECMP routing protocol for fat-tree enabled data centre networks," in *International Conference on Ubiquitous Information Management and Communication*, 2021, pp. 1–8.

[16] Y. C. Wang and T. J. Hsiao, "URBM: User-rank-based management of flows in data center networks through SDN," in *IEEE International Conference on Computer Communication and the Internet*, 2022, pp. 142–149.

[17] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 640–656, 2017.

[18] Y. C. Wang and H. Hu, "An adaptive broadcast and multicast traffic cutting framework to improve Ethernet efficiency by SDN," *Journal of Information Science and Engineering*, vol. 35, no. 2, pp. 375–392, 2019.

[19] R. Wazirali, R. Ahmad, and S. Alhiyari, "SDN-OpenFlow topology discovery: An overview of performance issues," *Applied Sciences*, vol. 11, no. 15, pp. 1–30, 2021.

[20] Y. Ma, N. Zhang, and J. Li, "Improved sequential least squares programming–Driven feasible path algorithm for process optimisation," *Computer Aided Chemical Engineering*, vol. 51, pp. 1279–1284, 2022.

[21] Y. C. Wang and Y. C. Wang, "Efficient and low-cost defense against distributed denial-of-service attacks in SDN-based networks," *International Journal of Communication Systems*, vol. 33, no. 14, pp. 1–24, 2020.

[22] Mininet. [Online]. Available: http://mininet.org

[23] Ryu. [Online]. Available: https://ryu-sdn.org

[24] Open vSwitch. [Online]. Available: https://www.openvswitch.org

[25] iPerf. [Online]. Available: https://iperf.fr

[26] Y. C. Wang, "A two-phase dispatch heuristic to schedule the movement of multi-attribute mobile sensors in a hybrid wireless sensor network," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 709–722, 2014.

[27] Y. C. Wang and L. C. Yen, "Collaborative route management to mitigate congestion in multi-domain networks using SDN," in *IEEE Annual Computing and Communication Workshop and Conference*, 2022, pp. 988–994.