Flexible Flow Converging: A Systematic Case Study on Forwarding Plane Programmability of Protocol-Oblivious Forwarding (POF)

Daoyun Hu, Shengru Li, Huibai Huang, Wenjian Fang, Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—It is known that software-defined networking (SDN) can support effective traffic engineering (TE) with the global view of networks. Hence, OpenFlow was used to realize flow-based TE. However, there is a tradeoff between the volume of installed flow entries and the granularity of TE. Moreover, the protocol-dependent nature of OpenFlow might limit the flexibility and adaptivity of TE. In this work, we leverage the protocol-oblivious forwarding (POF) technology that can make each switch work as a protocol-independent white box, and utilize its forwarding plane programmability to design a novel flexible flow converging (F-FC) scheme for realizing SDN-based fine-grained TE. Specifically, we design both the network system and the F-FC algorithms running on it and conduct experiments to demonstrate that our proposed scheme can not only reduce the volume of installed flow entries in switches, but also realize fine-grained TE to achieve better utilization on network bandwidth.**

***Index Terms*—Software defined networking (SDN), Protocol-oblivious forwarding (POF), Flexible flow converging (F-FC).**

I. Introduction

With the fast development of the Internet, new applications are emerging quickly and their Quality-of-Service (QoS) requirements on latency, jitter, *etc* becomes stricter and stricter. Therefore, the fine-grained traffic engineering (TE) that can balance network utilization well to support these QoS requirements attracts intensive attentions. Meanwhile, it is known that by leveraging centralized network control and management (NC&M), software-defined networking (SDN) opens up a new paradigm to optimize TE flexibly based on the global view of networks [1, 2]. As a well-known protocol to support SDN, OpenFlow [3, 4] has standardized the south/north-bound communications between the control and forwarding planes for flow-level traffic control. Hence, with OpenFlow, one can realize per-flow based TE by manipulating the flow entries on switches. This means that dedicated flow entries need to be stored and maintained for each flow in ternary content addressable memory (TCAM). However, as TCAM is expensive, SDN switches usually can only store a very limited number of flow entries with it [5]. On the other hand, per-flow based TE with OpenFlow makes the control and forwarding planes interact frequently to adapt to the dynamics of each flow, which might exhaust the processing capacity of both OpenFlow controller and switches quickly [6]. Hence, it is desired to develop SDN-based fine-grained TE techniques that can relieve the aforementioned scalability issues.

Previously, people have considered the schemes with either forwarding equivalence class (FEC) [7] or source routing [8] to address the scalability issues related to flow entries. With FEC, flows on the same forwarding path can be categorized into one FEC and thus share the same flow entries. Source routing encapsulates the forwarding path directly into packets, *i.e.*, sequentially encoding the designated output port of each switch along the path as labels in packets' headers, and then makes switches pop the corresponding label in each hop to forward packets correctly. As source routing packets carry complete routing information, there is no need to store it in switches as flow entries. Hence, the number of flow entries can be greatly reduced.

However, since the implementations of FEC and source routing in OpenFlow networks generally leverage the multi-protocol label switching (MPLS) labels, a few issues might arise. For instance, the principle of MPLS labeling can lead to coarse granularity of TE with FEC and the fixed length of MPLS labels restricts the flexibility of source routing. Basically, these issues are caused by the protocol-dependent nature of the OpenFlow. OpenFlow is built based on legacy protocols, or in other words, it defines the matching fields in flow tables according to existing network protocols. This is the reason why we need to update the matching fields and corresponding actions constantly, *i.e.*, the latest OpenFlow v1.5 supports 44 matching fields and 20 actions while the numbers in OpenFlow v1.0 were 12 and 10. Nevertheless, even with all these updates, OpenFlow still cannot support new protocols such as FEC and source routing seamlessly and efficiently.

Recently, a few new SDN approaches have been proposed to enhance the programmability of forwarding plane for making it protocol-independent, *e.g.*, protocol oblivious forwarding (POF) [9, 10] and programming protocol independent packet processors (P4) [11]. P4 supports a high-level language for programming packet processors, which defines an open and flexible interface to process packets. While POF provides a protocol-independent instruction set that makes each forwarding element a white box. Specifically, POF defines flow table search keys and packet fields as $<$*offset, length*$>$ tuples, and thus a forwarding element can parse and process packets based on the tuples without thinking about the network protocols in advance. Hence, POF networks can support new network protocols on demand. Both P4 and POF are considered in the protocol-independent forwarding (PIF) project [12] proposed

D. Hu, S. Li, H. Huang, W. Fang and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received July 11, 2016.

by the open networking foundation (ONF).

In this paper, we propose a novel flexible flow converging (F-FC) scheme based on POF to realize SDN-based fine-grained TE and utilize the scheme to perform a systematic case study on the forwarding plane programmability of POF. Specifically, we design both the network system and the F-FC algorithms for it and conduct experiments to demonstrate that our proposed scheme can not only reduce the volume of installed flow entries in switches, but also realize fine-grained TE to achieve better utilization on network bandwidth. The major contributions of this paper are listed as follows.

a) We propose a POF-based F-FC scheme, which can reduce the installed flow entries in switches and support fine-grained TE simultaneously.
b) We formulate an integer linear programming (ILP) model for the algorithm design of F-FC and prove the $\mathcal{NP}$-hardness of the problem. Then, we design two time-efficient heuristic algorithms and use numerical simulations to verify that they can provide near-optimal F-FC solutions.
c) We design and implement the POF-based network system that can realize the F-FC algorithms.

The rest of this paper is organized as follows. Section II provides a brief survey on the related work. We describe the framework of POF-based F-FC in Section III. The ILP model for F-FC and related complexity analysis are included in Section IV and the heuristic algorithms are designed in Section V. Section VI uses numerical simulations to evaluate the proposed F-FC algorithms and we show the experimental demonstrations and results in Section VII. Finally, Section VIII summarizes the paper.

II. Related Work

In order to achieve QoS guarantees in SDN networks, the studies in [13] proposed to assign dedicated flow entries for flows with specific bandwidth requirements. In [14], we proposed to use IP-forwarding interchanging enabled by OpenFlow to realize QoS-aware flexible TE in a hybrid network where IPv4 and IPv6 coexist. Then, in [15, 16], we investigated QoS-aware video streaming with the assistance of SDN. However, these studies did not consider the fact that the installed flow entries are limited in a practical SDN switch. In [17], the authors considered the restriction on installed flow entries and proposed an approximation algorithm to maximize the throughput of SDN networks. Specifically, they assumed that flows with the same source and destination could be forwarded over K paths and a dedicated flow entry is assigned to each path. Lee *et al.* [18] proposed a load-balancing algorithm for SDN networks, which also utilized K-shortest paths and assigned dedicated flow entries to flows. Nevertheless, the work in [17, 18] did not try to use flow converging scheme to reduce installed flow entries.

Meanwhile, previous studies have also considered to reduce the volume of installed flow entries in SDN switches. In [19], the authors proposed to decompose a large flow table into equivalent sub-tables such that multiple flows can share the sub-tables to improve the efficiency of flow table utilization. However, making multiple flows share same sub-tables can limit the granularity of TE. Another way to reduce installed flow entries is to compress the flow tables. Nevertheless, the work in [20] approved that it is an $\mathcal{NP}$-complete problem to compress flow tables even when they are two-dimensional. Note that, most of the flow tables in OpenFlow networks are multi-dimensional. Hu *et al.* [21] developed a flow converging scheme that categorizes flows sharing the same path segment into an FEC and utilizes a common label to identify it. However, the scheme needs to precalculate K shortest paths for each source-destination pair in the topology, which might become difficult to implement for large scale networks and also limit the flexibility of TE.

The implementation of source routing with OpenFlow can also reduce the volume of installed flow entries [8, 22, 23]. Nevertheless, the OpenFlow-based source routing is still protocol-dependent, which means that the definition of the output port related matching fields still needs to comply with the existing protocols (*e.g.*, MPLS or VLAN) and cannot be adjusted adaptively for each network. Moreover, the additional overhead due to source routing could be another issue. For instance, a flow with 5 hops may require five MPLS labels to encapsulate the routing information, which means that the header length of each packet would increase 160 bits. To reduce this overhead, segment routing was designed as a modified version of source routing, which assign a label to a multi-hop path segment [24]. The work in [25] studied how to realize TE with segment routing, but it did not consider the constraint from the limited flow entries that can be installed on each switch. Basically, for segment routing, the tradeoff between the volume of installed flow entries and packet overhead should be addressed carefully, and when TE is considered, we also need to think about the bandwidth constraint on network links. With these considerations, we have designed a POF-based F-FC system for fine-grained TE and showed some preliminary results in [26]. In this work, we redesign the related algorithm and protocol to make the system operate in a more efficient way and conduct more theoretical and experimental analysis to make the study more comprehensive.

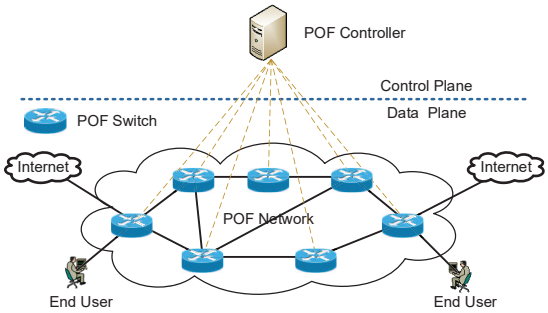
III. POF-based Flexible Flow Converging (F-FC)

In this section, we briefly introduce the operation principle of POF and explain our design of POF-based F-FC.

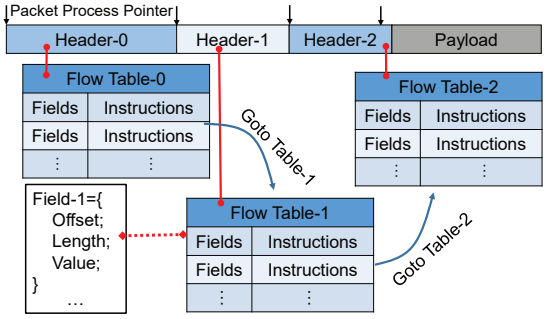
A. Protocol Oblivious Forwarding (POF)

As shown in Fig. 1(a), POF networks just take the form of standard SDN network architecture, which use a centralized controller in the control plane to manage the switches in the data plane. Compared with OpenFlow, the major innovation of POF is that it aims to make the switches work as white boxes such that packet forwarding procedure in them is protocol-independent. Fig. 1(b) explains the packet forwarding procedure in POF, which is realized with protocol-independent matching fields and flow instruction sets (FIS).

- **Matching Fields**: The search key of a matching field in POF is defined as <*offset, length*> tuple, where *offset* indicates the start location of the field in a packet and *length* provides the length of field in bits [9].

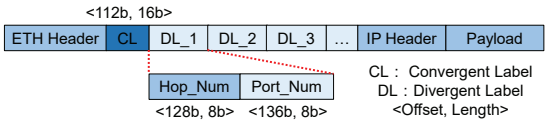


(a) Network architecture of POF.

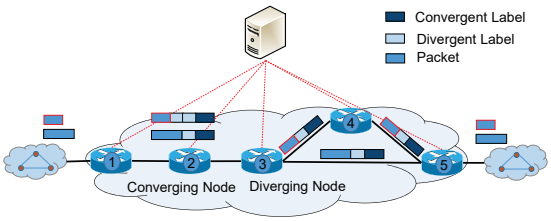


(b) Packet forwarding procedure in POF switches.

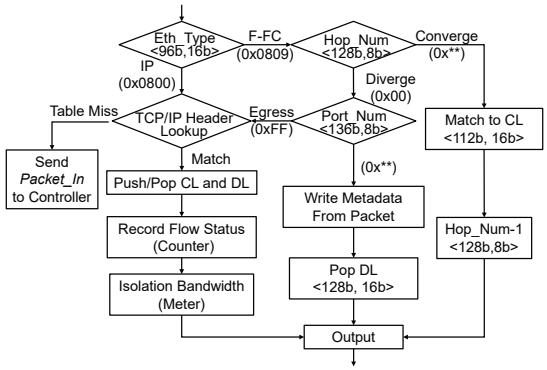
Fig. 1. Overview of POF.



(a) Packet format defined for F-FC.



(b) Operation principle of POF-based F-FC.



(c) POF-based flow processing procedure for F-FC.

Fig. 2. Overview of POF-based F-FC.

- **POF-FIS**: Instructions defined in POF use the <*offset*, *length*> tuples to locate fields in a packet to operate on [10]. For example, if we want to insert a new field into the packet, we can utilize the *Add_Field* instruction in POF-FIS to implement it easily.

B. Protocol Design for F-FC

In F-FC, we merge the flows to the same destination into one converged flow and assign a convergent label (CL) to it. To guarantee fine-grained TE, we also assign one or multiple divergent labels (DLs) to each individual flow in the converged flow. Basically, the number of DLs in a packet depends on the number of diverging nodes, *i.e.*, the nodes on which the converged flow can split, on its routing path. Then, the POF-based F-FC packets use the format as shown in Fig. 2(a). Here, for an Ethernet frame, the <*offset*, *length*> tuple of CL is <112, 16>, which means that CL is inserted directly after the Ethernet header and its length is 16 bits. Each DL contains two fields, where *Hop_Num* indicates the hop-count from the current node to the next diverging node and *Port_Num* specifies the designated output port for the packet on the next diverging node. Here, the lengths of *Hop_Num* and *Port_Num* are both 8 bits. It can be seen that F-FC is similar as source routing and segment routing. But the overhead caused by F-FC is smaller than that of source routing because DLs are only encapsulated for the diverging nodes rather than every hop on the routing path. Meanwhile, in terms of TE, F-FC is more flexible than segment routing since with DL, we can easily adjust the routing path of each individual flow.

Fig. 2(b) gives an intuitive example to explain the operation principle of POF-based F-FC. When the packets belonging to two flows arrive at the ingress switch of the POF network, they get transformed into the F-FC format. Then, at *Node* 2, the switch forwards them as a converged flow by examining their first DLs and then matching to their CLs. Next, at *Node* 3, the two flows need to be split to avoid congestion, and the switch knows this by checking their first DLs and finding that the *Hop_Num* fields in them are 0. Hence, the switch pops the first DLs in the packets and forwards the packets to the output ports encoded in the *Port_Num* fields in the DLs. On *Node* 4, the switch forwards the packets of one flow by examining their first DLs and then matching to their CLs. When the packets reach the egress switch on *Node* 5, they are converted back to the format in the legacy network. Note that, thanks to the protocol-independent nature of POF, the lengths of CL and DL can be adaptive with the network status, *e.g.*, network size, number of ingress-egress pairs, and maximum number of output ports on switches, even though they are both defined as 16 bits here.

Fig. 2(c) shows the flow processing procedure of POF-based F-FC. When a packet arrives at a POF switch, the switch first checks the type of the packet. If it is a conventional IP packet (*i.e.*, the *Eth_Type* field is 0x0800), the switch will record its flow status and push corresponding CL and DL into its header (*i.e.*, converting the packet into the F-FC format) after checking its IP and TCP/UDP headers. Otherwise, if the packet is an F-FC one (*i.e.*, the *Eth_Type* field is 0x0809), the switch checks whether the *Hop_Num* field in the first DL is 0. If yes, the switch is on a diverging node of the flow, and the packet should be forwarded to the output port encoded in the

Port_Num field in the first DL. Here, we just use one flow entry on each switch to match and process all the diverging flows, by leveraging the metadata memory and *write-metadata-from-packet* instruction defined in POF [27]. Therefore, we not only reduce the required flow entries but also minimize the interactions between the controller and switches. Then, the first DL is popped from the packet. Note that, for the special case when the switch is the egress one to a legacy network, the *Port_Num* field in the first DL would be 0xFF. Then, the switch just pops the CL and DL and processes it as a conventional IP packet. On the other hand, if the *Hop_Num* field in the first DL is not 0, which means that the switch is not on a diverging node of the flow, the packet is forwarded by matching to the CL and the *Hop_Num* field in the first DL is reduced by 1.

Our implementation leverages the multi-table scheme to realize the flow processing discussed above. Specifically, the flow entries in the tables for handling *Eth_Type*, *Hop_Num*, and *Port_Num* can be installed in switches in advance, since these flow entries are common and can be shared by all the flows. Hence, their volume is much smaller than that of the flow entries in the tables for looking up TCP/IP headers and handling CL. Moreover, with F-FC, flows in a converged flow are processed by the same flow entry on converging nodes, which helps to reduce installed flow entries further. Meanwhile, as each individual flow in a converged flow can be split and forwarded independently based on their DLs, the F-FC scheme ensures good performance for fine-grained TE.

IV. Theoretical Model for F-FC Algorithm Design

In this section, we describe the network model for F-FC, formulate an ILP model to maximize the network operator's revenue from serving the flows with F-FC, and analyze the complexity of the problem with the ILP model.

A. Network Model

We denote the topology of the POF network as a directed graph $G(V, E, \Gamma, C)$, where V and E are the sets of nodes and links, respectively. The number of available flow entries on node $v \in V$ is represented as $\gamma_v \in \Gamma$, while the bandwidth capacity of link $e \in E$ is $c_e \in C$. If link $e \in E$ is from u to v, we denotes it as $e = (u, v)$. The set of flow requests is Q, and the i-th request is denoted as $q_i = \{s_i, d_i, b_i, h_i\} \in Q$, where i is its index, s_i and d_i are its ingress and egress node to the POF network, b_i is the bandwidth requirement, and h_i is its holding time. By serving a flow request q_i, the network operator can obtain a revenue as

$$\omega_i = \lambda \cdot b_i \cdot h_i, \tag{1}$$

where λ is the constant revenue coefficient. The objective of F-FC is to maximize the revenue of the operator under the condition that both the number of available flow entries on switches and the bandwidth on links are limited.

B. ILP Model for F-FC

Parameters:

- $c_{(u,v)}$: the bandwidth capacity of link (u, v).
- $q_i = \{s_i, d_i, b_i, h_i\}$: the i-th flow request arriving at the POF network.
- γ_v: the number of available flow entries on node v.

Variables:

- x_i: the boolean variable that equals 1 if a feasible path with enough bandwidth and flow entries is found for flow request q_i, and 0 otherwise.
- $f^i_{(u,v)}$: the boolean variable that equals 1 if link (u, v) is used by q_i, and 0 otherwise.
- l^i_v: the integer auxiliary variable for ensuring that there is no loop on the routing path of q_i.
- θ^d_v: the boolean variable that equals 1 if node v is used by the flows whose destination is d, and 0 otherwise.
- R: the integer variable that indicates the total revenue from serving the flows.

Objective:

The optimization objective is to maximize the operator's total revenue from serving the flows with F-FC, as

$$Maximize \quad R = \sum_{i=1}^{|Q|} x_i \cdot \omega_i. \tag{2}$$

Constraints:

(a) *Flow Conservation Constraints*:

$$\sum_{\{v:(u,v)\in E\}} f^i_{(u,v)} - \sum_{\{v:(v,u)\in E\}} f^i_{(v,u)} = \begin{cases} x_i, & u = s_i, \\ -x_i, & u = d_i, \\ 0, & \text{otherwise}, \end{cases} \quad \forall i. \tag{3}$$

Eq. (3) ensures that if a flow q_i is served, one and only one path is built for it, otherwise, no path is built.

(b) *Loop Avoidance Constraints*:

$$l^i_u - l^i_v + |V| \cdot f^i_{(u,v)} \le |V| - 1, \quad \forall i,\ \forall (u, v) \in E. \tag{4}$$

Eq. (4) ensures that the routing path of each flow is loopless, where $|V|$ indicates the total number of nodes in V.

(c) *Bandwidth Constraints*:

$$\sum_{i=1}^{|Q|} f^i_{(u,v)} \cdot b_i \le c_{(u,v)}, \quad \forall (u, v) \in E. \tag{5}$$

Eq. (5) ensures that the bandwidth occupied by the flows on each link does not exceed its capacity.

(d) *Flow Converging Constraints*:

$$|Q| \cdot \theta^d_v \ge \left(\sum_{\{i:d_i=d, s_i\ne v\}} \sum_{\{u:(v,u)\in E\}} f^i_{(v,u)} \right), \ \forall v, d \in V, \tag{6}$$

$$\theta^d_v \le \sum_{\{i:d_i=d, s_i\ne v\}} \sum_{\{u:(v,u)\in E\}} f^i_{(v,u)}, \quad \forall v, d \in V. \tag{7}$$

Eqs. (6)-(7) ensure that flows with the same destination can be merged into one converged flow on an intermediate node.

(e) *Flow Entry Constraints*:

$$\sum_{d\in V} \theta^d_v + \sum_{\{i:s_i=v \| d_i=v\}} x_i \le \gamma_v, \quad \forall v \in V. \tag{8}$$

Eq. (8) ensures that the number of flow entries installed on node v does not exceed the number of available flow entries

γ_v there. Note that, each flow needs a dedicated flow entry on its ingress and egress switches to record the flow status, while on the intermediate switches, flows in the same converged flow can share one flow entry for handling CL. Hence, the number of installed flow entries on node v equals the summation of the number of converged flows using it and the number of individual flows whose ingress or egress node is it[1].

C. Complexity Analysis

The complexity of an ILP model can usually be estimated by looking into the number of variables and constraints used in it. In the ILP model formulated above, the number of variables is $|Q|\cdot(|E|+|V|+1)+|V|^2$, while the number of constraints is upper-bounded by $(|Q|+1)\cdot(|E|+|V|)+2\cdot|V|^2$. We analyze the complexity of the problem more formally as follows.

***Theorem** 1:* The optimization described by the ILP model in Subsection IV-B is $\mathcal{NP}$-hard.

Proof: For the optimization, if we assume that $\gamma_v = +\infty,\ \forall v \in V$ and make the flow entry constraints irrelevant, we can transform this special case of our problem into a general case of the unsplittable flow problem (UFP) [28]. Then, if we assume that $c_{(u,v)} = 1,\ \forall (u,v) \in E$ and $\omega_i = 1,\ \forall i$, the goal of UFP simply becomes to maximize the number of link-disjoint paths between the source-destination pairs in $G(V,E)$, which is the well-known maximum edge-disjoint paths (EDP) problem [29]. As proven in [29], EDP is $\mathcal{NP}$-hard. Hence, we can see that UFP is also $\mathcal{NP}$-hard and so does our optimization problem. ■

V. Heuristic Algorithms for F-FC

In this section, we design two time-efficient heuristics to solve the optimization described above quickly such that the F-FC system can practically implement them in a POF controller for on-line flow provisioning in dynamic network environment.

A. Two-stage Heuristic Algorithm (T-HA)

We first consider an F-FC algorithm that processes the routing path, bandwidth and flow entry allocation in two stages. Specifically, we calculate K least weighted paths for each flow request based on the current network status, and then choose the path that would cause the smallest flow entry increase to provision the flow. *Algorithm* 1 shows the detailed procedure of the two-stage heuristic algorithm (T-HA). *Lines* 2-3 are for the initialization. Here, we introduce an auxiliary topology $G^a(V^a, E^a, \Gamma^a, C^a)$ to help the POF controller determine on which paths q_i can be served with sufficient bandwidth and flow entries. G^a is initialized as G. Then, for each node $v \in V^a$, we define

$$z_v = \begin{cases} +\infty, & \theta_v^{d_i} = 0 \text{ and } \gamma_v = 0, \\ 0, & \text{otherwise}, \end{cases} \quad \forall v \in V^a. \tag{9}$$

where $\theta_v^{d_i} \in \Theta$ denotes the status of the converged flow to destination d_i on node v, *i.e.*, if there is no converged flow to d_i on node v, we have $\theta_v^{d_i} = 0$, and $\theta_v^{d_i} = 1$ otherwise. If $\theta_v^{d_i} = 0$, we need to install a new flow entry for q_i on node v. Hence, if there is no available flow entry on v (*i.e.*, $\gamma_v = 0$) and $\theta_v^{d_i} = 0$, Eq. (9) sets the cost of using v to provision q_i as $z_v = +\infty$. Next, we consider bandwidth resources and define the cost of using a link (u,v) to provision q_i as

$$m_{(u,v)} = \begin{cases} +\infty, & b_i > c_{(u,v)}, \\ 1, & \text{otherwise}, \end{cases} \quad \forall (u,v) \in E^a. \tag{10}$$

Basically, if the bandwidth requirement of q_i is larger than the available bandwidth on link (u,v), the cost of using the link should be set as $+\infty$, and 1 otherwise. Finally, the total cost of using link (u,v) and its two end-nodes is

$$\delta_{(u,v)} = m_{(u,v)} + z_u + z_v. \quad \forall (u,v) \in E^a. \tag{11}$$

Line 4 updates the auxiliary topology G^a with Eqs. (9)-(11) to include the node and link costs defined above. Then, *Lines* 5-9 try to calculate K least weighted paths in G^a as the path candidates to provision q_i. If no path can be obtained, either bandwidth resources or flow entry resources or both are insufficient and we just drop the flow request. *Lines* 10-13 try to provision q_i on each path candidate p_k and store the corresponding flow entry increase in η_k. In *Line* 14, we provision q_i with the path that would result in the smallest flow entry increase. At last, we update the matrix of flow converging Θ and the network topology G in *Lines* 15-16.

The time complexity of T-HA mainly depends on the method for calculating the K least weighted paths. Here, we leverage the algorithm developed in [30], whose time complexity is $O(K\cdot|V|\cdot(|E|+|V|\cdot\log|V|))$. Hence, the time complexity of T-HA is $O(K\cdot|Q|\cdot|V|\cdot(|E|+|V|\cdot\log|V|))$.

Algorithm 1: Two-stage Heuristic Algorithm

```
1  for each request q_i ∈ Q do
2      obtain information of q_i = {s_i, d_i, b_i, h_i};
3      P = ∅, G^a(V^a, E^a, Γ^a, C^a) = G(V, E, Γ, C);
4      update status of auxiliary topology G^a with Eqs. (9)-(11);
5      calculate K least weighted paths in G^a for s_i-d_i to store in P;
6      if P = ∅ then
7          drop request q_i;
8          continue;
9      end
10     for each k ∈ [1, K] do
11         apply F-FC scheme for q_i on path p_k ∈ P;
12         store flow entry increase on p_k in η_k;
13     end
14     provision q_i using the p_k with the smallest η_k;
15     update the matrix of flow converging Θ;
16     update network topology G(V, E, Γ, C);
17 end
```

B. Integrated Heuristic Algorithm (I-HA)

It is known that online TE should try to distribute traffic load as balanced as possible such that random traffic demands can

[1]Note that, as the flow entries for handling *Eth_Type*, *Hop_Num*, and *Port_Num* are common and can be shared by all the flows, their volume is much smaller and can be ignored here.

be easily served in the future. This suggests that the differentiated bandwidth capacities on links in the POF network should be addressed carefully in F-FC. Therefore, we propose an integrated heuristic algorithm (I-HA) that performs F-FC with the joint consideration of bandwidth and flow entry resources.

We first redesign $\delta_{(u,v)}$, *i.e.*, the cost of using a link (u,v) to provision q_i, with the following equations.

$$z_v = \begin{cases} 0, & \theta_v^{d_i} = 1, \\ +\infty, & \theta_v^{d_i} = 0 \text{ and } \gamma_v = 0, \\ \dfrac{\widehat{\gamma_v}}{\gamma_v}, & \text{otherwise}, \end{cases} \quad \forall v \in V^a, \tag{12}$$

where $\widehat{\gamma_v}$ represents the initial available flow entries on node v when the POF network is empty.

$$z_{(u,v)} = \frac{1}{2}(z_u + z_v), \quad \forall (u,v) \in E^a, \tag{13}$$

$$m_{(u,v)} = \begin{cases} +\infty, & b_i > c_{(u,v)}, \\ \dfrac{\widehat{c_{(u,v)}}}{c_{(u,v)}}, & \text{otherwise}, \end{cases} \quad \forall (u,v) \in E^a, \tag{14}$$

where $\widehat{c_{(u,v)}}$ denotes the initial available bandwidth on link (u,v) when the POF network is empty.

$$\alpha = 1 - \frac{\frac{1}{|V^a|}\sum\limits_{v \in V^a} \frac{\gamma_v}{\widehat{\gamma_v}}}{\left(\frac{1}{|V^a|}\sum\limits_{v \in V^a} \frac{\gamma_v}{\widehat{\gamma_v}}\right) + \left(\frac{1}{|E^a|}\sum\limits_{(u,v) \in E^a} \frac{c_{(u,v)}}{\widehat{c_{(u,v)}}}\right)}, \tag{15}$$

where α is the coefficient for balancing the importance of bandwidth and flow entry resources. Here, we try to be cautious on the resources that are becoming insufficient such that the flows would not be provisioned on the congested hot spots in the network.

$$\delta_{(u,v)} = \alpha \cdot z_{(u,v)} + (1-\alpha) \cdot m_{(u,v)}. \tag{16}$$

Then, *Algorithm* 2 shows the detailed procedure of I-HA. *Lines* 2-3 are for the initialization, and *Line* 4 updates the cost of using a link (u,v) to provision q_i with Eqs. (12)-(15) in the auxiliary topology G^a. We use *Lines* 5-9 to find a suitable path (*i.e.*, the least weighted one) for provisioning q_i. If no path can be found, the flow request would be dropped. Otherwise, we provision q_i on the path as shown in *Line* 10. At last, *Lines* 11-12 update the network status. In this algorithm, we calculate the least weighted path with the Djkstra algorithm, and thus the overall time complexity is $O(|Q| \cdot (|E| + |V| \cdot \log |V|))$.

Algorithm 2: Integrated Heuristic Algorithm

```
1  for each request q_i ∈ Q do
2      obtain information of q_i = {s_i, d_i, b_i, h_i};
3      P = ∅, G^a(V^a, E^a, Γ^a, C^a) = G(V, E, Γ, C);
4      update status of auxiliary topology G^a with Eqs.
       (12)-(15);
5      calculate the least weighted path p in G^a for
       s_i-d_i;
6      if no path can be found then
7          drop request q_i;
8          continue;
9      end
10     provision q_i using path p;
11     update the matrix of flow converging Θ;
12     update network topology G(V, E, Γ, C);
13 end
```

VI. Numerical Simulations

In this section, we perform numerical simulations to evaluate the performance of the algorithms discussed in Sections IV and V. We implement the ILP model in Lingo v11.0 and simulate the proposed heuristic algorithms with MATLAB R2013, and all the simulations run on a Windows server with 2.2 GHz CPU and 32 GB RAM. We perform two types of simulations, *i.e.*, one-time operation and dynamic operation. The former one only tries to provision a set of requests and then stops, while the latter one conducts dynamical simulations to consider the case in which flow requests can come and leave on-the-fly. Considering the time complexity of the ILP model, we only simulate it in a small-scale network with one time operation. While the proposed heuristics are simulated in both the small-scale network and a large-scale network with a 100-node random topology. For the benchmark algorithms, we introduce two: one is flow provisioning without flow converging (Without-FC) and the other is X-Path. For Without-FC, each flow is provisioned by installing a dedicated flow entry on each node on its routing path, which is similar to the per-hop configuration scheme in conventional OpenFlow. Here, for fair comparisons, the routing paths are calculated with the scheme proposed in I-HA, *i.e.*, obtaining the least weighted paths in the auxiliary topology. For X-Path, we use the K shortest path algorithm to calculate the path candidates, and then flows with the same ingress-egress pair can be converged on the path candidates for reducing installed flow entries. Note that, X-Path is similar to FEC and it assigns one dedicated label to each path candidate.

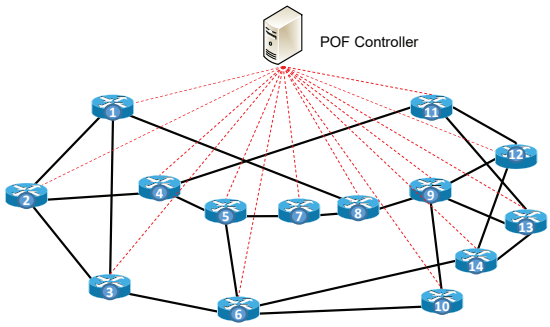


Fig. 3. Topology used in simulations for one-time operation.

A. One-Time Operation

The simulations for one-time operation use the small-scale topology in Fig. 3, which consists of 14 nodes and 21 links. The initial bandwidth capacity on each link is set as 10 Mbps, while the initial flow entry capacity on each node is 15. The bandwidth requirement of each flow is 1 Mbps and its ingress and egress nodes are selected randomly. Table I shows the simulation parameters.

TABLE I
SIMULATION PARAMETERS

Parameters	One-Time	Dynamic (Flow Entry Constrained)	Dynamic (Bandwidth Constrained)
$\widehat{c_{(u,v)}}$	10	1000	$[50, 70]$
$\widehat{\gamma_v}$	15	$[350, 450]$	1000

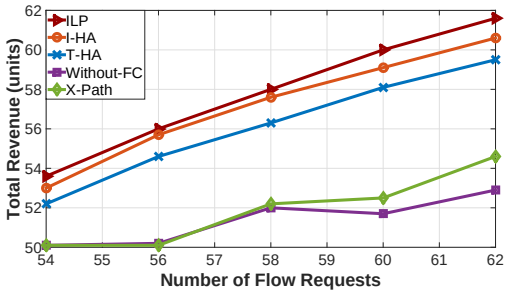
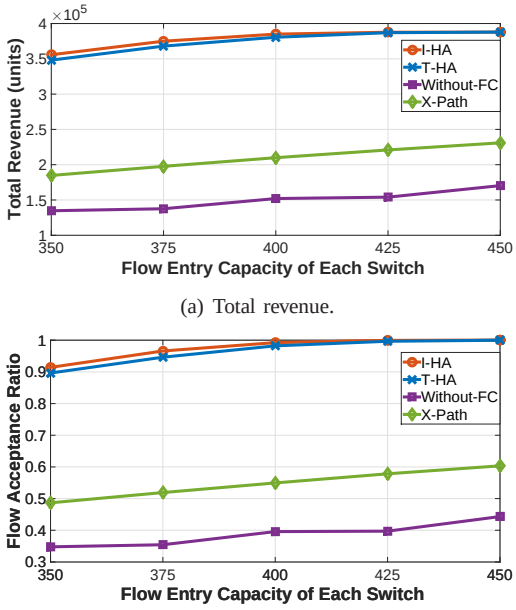


Fig. 4. Simulation results on total revenue for one-time operation.

Fig. 4 shows the results on total revenue from serving the flow requests. The total revenue is calculated with Eq. (2) and each data point in Fig. 4 is obtained by averaging the results from 10 independent simulations. As expected, ILP obtains the highest total revenue among the five algorithms. For the two proposed heuristics, I-HA provides higher total revenue than T-HA, and its results are very close to those from ILP. By contrast, the total revenues from the benchmarks (*i.e.*, Without-FC and X-Path) are much smaller than our proposed algorithms. The average running time of the algorithms are listed in Table II. It can be seen that ILP takes real long time to get the results, which makes it impractical for being implemented in a POF controller for online flow provisioning. Both T-HA and I-HA run reasonably fast and I-HA runs significantly faster than T-HA. Therefore, the simulation results for one-time operation suggest that I-HA is a promising algorithm for realizing F-FC in POF networks, as it provides higher total revenue with shorter computation time.

B. Dynamic Operation

The simulations for dynamic operation use a 100-node random topology that is generated with the method developed in [31]. The dynamic flow requests are generated with the Poisson traffic model in which the requests' arrivals follow the Poisson process with an average arrival rate of ν and the holding time of each request follows the negative exponential distribution with an average of $\bar{h}$. Then, the traffic load can be quantified with $\bar{h} \cdot \nu$ in Erlangs. Similar to the work in [18], we consider two types of flows. The bandwidth requirements of the first type of flows range within $[0.1, 0.9]$ Mbps, and 80% of the total flows are in this type. The flows in the second type require $[1, 10]$ Mbps bandwidth capacity. The ingress and egress nodes of the flows are randomly chosen from the topology, and for T-HA and X-Path, we calculate $K = 5$ least weighted path candidates for each flow. The simulations consider two scenarios: 1) the bandwidth capacity is sufficient but the flow entry resources are constrained, and 2) the flow



(a) Total revenue.

(b) Flow acceptance ratio.

Fig. 5. Simulation results for dynamic operation with limited flow entries.

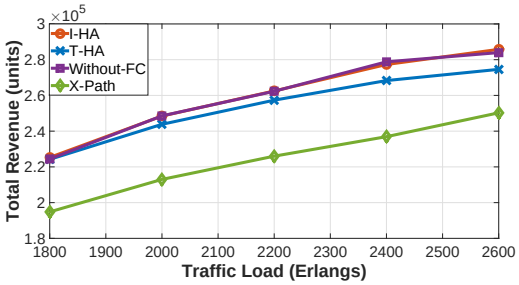
entry resources are sufficient but the bandwidth capacity is limited. Table I shows the simulation parameters.

Fig. 5 shows the results from the simulations for the first scenario. We observe that both the total revenue and the flow acceptance ratio increase with the flow entry capacity on each switch. This indicates that lack of flow entries can impact the POF network's performance significantly. Again, the results verify that T-HA and I-HA can achieve better performance than the two benchmarks. When comparing T-HA and I-HA, we can see that I-HA performs slightly better, in terms of both the total revenue and flow acceptance ratio. For the benchmarks, X-Path outperforms Without-FC since it makes flows with the same ingress-egress pairs share flow entries and can avoid the congestion due to limited flow entries.

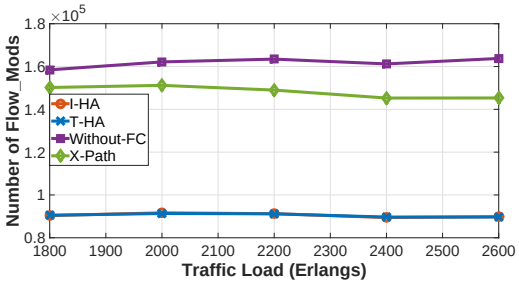
For the second scenario, we simulate 16000 flow requests, set the number of flow entries on each switch as 1000, and compare the algorithms' performance at different traffic loads. Fig. 6(a) plots the results on total revenue. It is interesting to notice that for this scenario, the total revenue from I-HA is almost the same as that from Without-FC. This is because as the flow entry resources are not the bottleneck any more, the granularity of TE becomes crucial for increasing the total revenue. In other words, since the bandwidth capacity is limited, if we can adjust the flows' routing paths most adaptively, the bandwidth can be utilized most efficiently and thus we can achieve the highest total revenue. Apparently, in this scenario, Without-FC performs the best for fine-grained

TABLE II
SIMULATION RESULTS ON AVERAGE RUNNING TIME (SECONDS)

# of Flow Requests	ILP	I-HA	T-HA	Without-FC	X-path
54	431.2	0.119	0.546	0.112	0.496
56	671.6	0.128	0.57	0.118	0.491
58	800.7	0.134	0.586	0.121	0.508
60	1030.6	0.136	0.621	0.122	0.525
62	1468.6	0.135	0.634	0.124	0.529



(a) Total revenue.



(b) Number of *Flow_Mod* messages.

Fig. 6. Simulation results for dynamic operation with limited bandwidth.

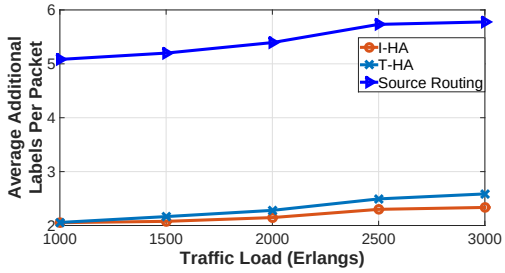


Fig. 7. Simulation results on packet overhead.

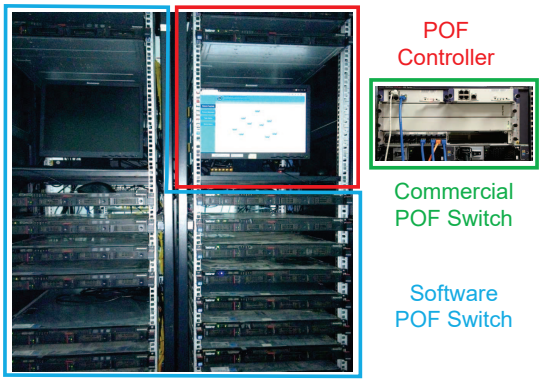


Fig. 8. Experimental testbed.

TE since it basically can adjust the routing path of each flow independently. Therefore, the fact that I-HA provides similar total revenue as Without-FC verifies that it can achieve fine-grained TE. Nevertheless, due to the fact that explicit routing makes an adverse effect on TE, X-Path provides the lowest revenue in this scenario.

Meanwhile, for the second scenario, we also collect the results on total number of *Flow_Mod* messages used for flow provisioning, as shown in Fig. 6(b). We observe that compared with the benchmarks, I-HA and T-HA send out much less *Flow_Mod* messages. This is because our F-FC scheme improves the reusage of flow entries, which reduce the number of *Flow_Mod* messages indirectly. However, since with the per-hop configuration scheme, Without-FC makes the controller send a *Flow_Mod* message for installing the flow entries of each flow, it uses the most *Flow_Mod* messages.

C. Comparison with Source Routing

We also conduct simulations to compare F-FC with source routing in terms of packet overhead. The simulations still use the 100-node topology, and we assume that both bandwidth and flow entry resources are sufficient. For source routing, an additional label is inserted into a packet to represent each hop. Fig. 7 shows the results on average additional labels per packet, which indicates that the overhead of F-FC is much less than that of source routing. These results further confirm the efficiency of our POF-based F-FC scheme. Note that, we do not compare F-FC to segment routing here. This is mainly because the performance of segment routing heavily depends on the method to obtain the path segments in the network [32]. Basically, to achieve fine-grained TE and adapt to dynamic traffic, the path segments and associated labels should be adjusted dynamically with a sophisticated algorithm.

VII. EXPERIMENT DEMONSTRATIONS

We then implement the F-FC scheme in a POF network system as shown in Fig. 8. Here, we have a POF controller and several POF switches, and the switches include both a commercial hardware switch (Huawei's NE40E-X3) and several software-based switches.

A. System Implementation

We extend the POF controller developed in [26] and implement the F-FC algorithms in it. Fig. 9 explains the procedure of flow forwarding with F-FC in the POF network.

- **Step 1**: A packet with QoS requirements arrives at an ingress switch of the POF network.

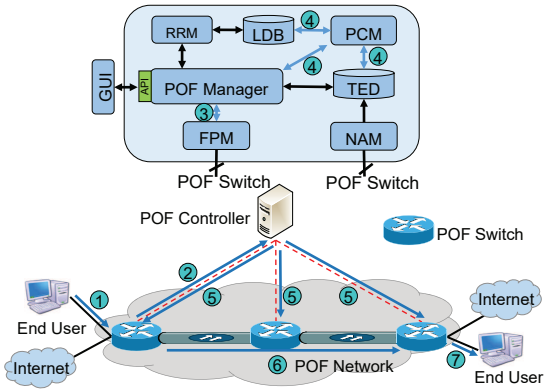


Fig. 9. Procedure for flow forwarding with F-FC in POF network.

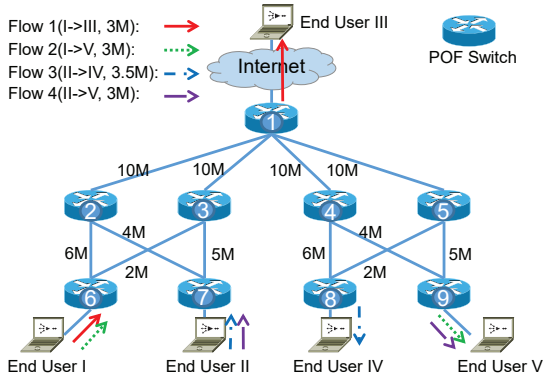


Fig. 10. Experimental topology for function verification.

- **Step 2**: There is no flow entry installed on the ingress switch for forwarding the packet, and thus the switch sends a *Packet-In* message to the POF controller.
- **Step 3**: The flow provision module (FPM) in the controller receives the *Packet_In* message and parses it for the flow's information. Then, FPM sends the information to POF Manager for path computation.
- **Step 4**: POF Manager asks the path computation module (PCM) to calculate the routing path based on the information from FPM and the current network status stored in the traffic engineering database (TED), and checks whether the flow can be served. If yes, corresponding CL and DL(s) are generated for the flow, and the label-flow mapping is stored in the label database (LDB). Otherwise, PCM will tell POF manager to drop the flow.
- **Step 5**: POF Manager generates flow entries and invokes FPM to encode them into *Flow_Mod* messages. FPM sends the messages to all the switches on the routing path to provision the flow.
- **Step 6**: The switches receive the *Flow_Mod* messages and install flow entries for the flow. At the ingress switch, packets are converted into the F-FC format and forwarded to the next hop based on the flow entry. Then, each intermediate switch processes the packets based on their CLs and first DLs.
- **Step 7**: At the egress switch, the packets are converted back to the format in the legacy network.

Note that, the network abstraction module (NAM) is responsible for collecting the topology information and storing them into TED, and it also contacts POF manager and TED when abnormal conditions are detected, *e.g.*, link down/up happens. The resource recycle module (RRM) checks the status of the flows periodically and recycles resources occupied by expired flows, *i.e.*, labels, flow entries and bandwidth.

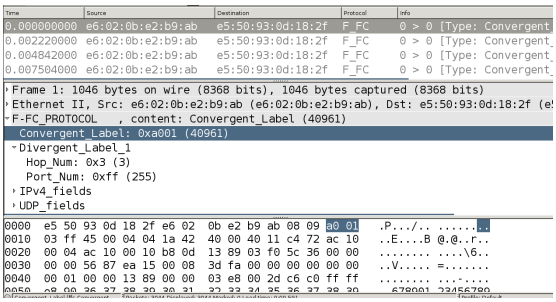
B. Experiments for Function Verification

In this experiment, we use the topology in Fig. 10 to verify the function of F-FC. Here, *Nodes* 2-9 in Fig. 10 are realized with running software-based POF switches on independent high-performance Linux servers, and *Node* 1 is the hardware POF switch that is based on Huawei's NE40E-X3. We have 5 end users connecting to the switches for sending or receiving packet flows. *End User* III connects to *Node* 1 through the Internet, while all the other end-users connect to their switches directly. Note that, the POF controller is also implemented and runs on a high-performance Linux server, which connects to all the POF switches directly. The bandwidth capacity on each link is shown in Fig. 10. Here, we assume that each switch except for the one on *Node* 1 can only accommodate two flow entries. The experimental scenario is as follows:

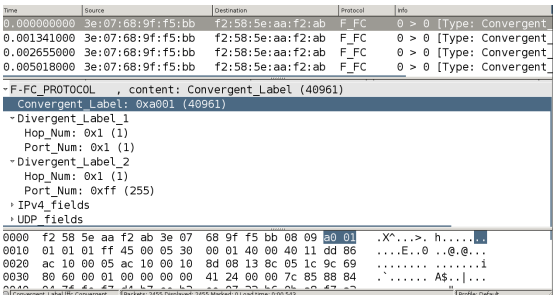
- **Step 1**: At $t = 0$ second, we start three packet flows. *Flow* 1 is from *End User* I to *End User* III and asks for 3 Mbps for file transfer, *Flow* 2 is from *End User* I to *End User* V and asks for 3 Mbps for file transfer, and *Flow* 3 is from *End User* II to *End User* IV and asks for 3.5 Mbps for the streaming of H.264 video sequence encoded as 1080P. The end users encapsulate all the flow packets in IPv4/UDP format and then send them out.
- **Step 2**: At $t = 26$ seconds, we have *Flow* 4 joined in, which is a new flow from *End User* II to *End User* V and asks for 3 Mbps for file transfer.

This experiment compares F-FC with I-HA and Without-FC, and their expected behaviors are as follows.

- **Without-FC**: In this scheme, each flow needs a dedicated flow entry. At $t = 0$ second, *Flow* 1 is forwarded on path 6→2→1. *Flows* 2 and 3 are forwarded with paths 6→2→1→5→9 and 7→3→1→4→8, respectively. Hence, each flow can be provisioned with enough bandwidth and flow entries. When *Flow* 4 tries to join in, the controller finds that there is no flow entry left on *Node* 2 even though the bandwidth on link 7→2 is sufficient. Hence, *Flow* 4 can only be routed on path 7→3→1→4→9. However, the bandwidth on link 7→3 is insufficient to accommodate *Flows* 3 and 4 simultaneously, and hence they will cause congestion.
- **F-FC with I-HA**: In this scheme, I-HA is implemented on the POF controller to enable it to converge flows with F-FC. At $t = 0$ second, *Flows* 1-3 are provisioned with the same routing scenario as with Without-FC. When *Flow* 4 tries to join in, the POF controller assigns it to path 7→2→1→4→9, where *Flows* 2 and 4 are converged on *Node* 2. Hence, *Node* 2 only needs two flow entries to forward *Flow* 1 and *Converged Flow* $(2, 4)$. Then, *Converged Flow* $(2, 4)$ can be split on *Node* 1 and



(a) Wireshark captures for Flow 2.



(b) Wireshark captures for Flow 4.

Fig. 11. Wireshark captures for F-FC packets.

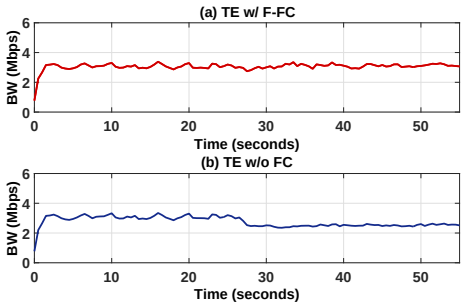


Fig. 12. Results on received bandwidth for video streaming.

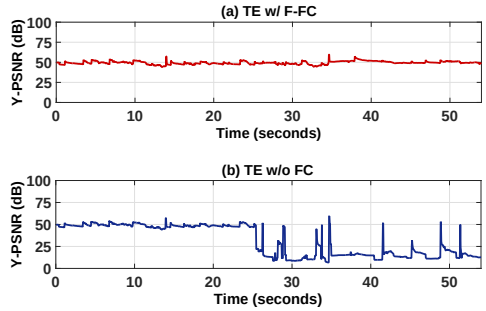
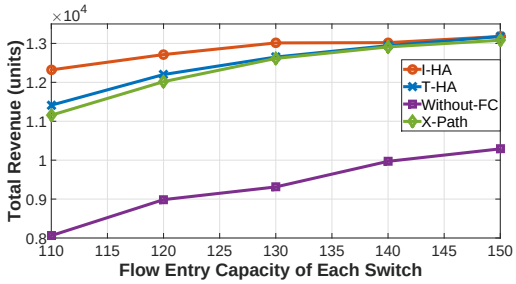
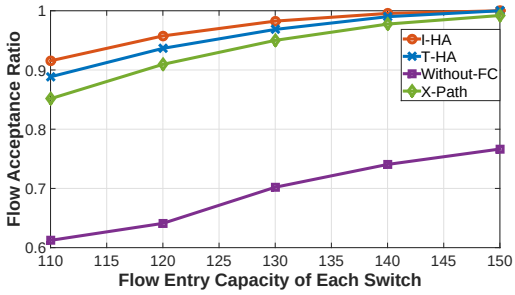


Fig. 13. Results on Y-PSNR of the received video.



(a) Total revenue.



(b) Flow acceptance ratio.

Fig. 14. Results for dynamic experiments with limited flow entries.

forwarded with path segments 1→4→9 and 1→5→9 for addressing the constrained bandwidth on link 5→9.

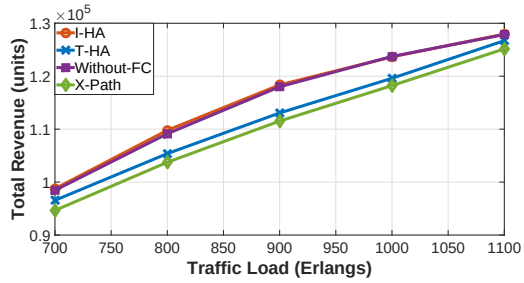
Fig. 11 shows the Wireshark captures for the F-FC packets of *Flows* 2 and 4 before *Node* 2. It can be seen that CLs and DLs are encapsulated into the packets of the two flows. Note that, the numbers of DLs used for the flows are different. For *Flow* 2, as it just uses the routing path of the converged flow end-to-end, the controller just assigns one DL to it for indicating the egress switch. On the other hand, the packets of *Flow* 4 have two DLs. Here, the first DL is for diverging *Flow* 4 from *Converged Flow* $(2, 4)$ on *Node* 1, and the second DL is used to point to the egress switch. Therefore, we can see that converging a new flow into an existing converged flow would not affect other in-service flows in the converged flow.

Fig. 12 plots the received bandwidth of *Flow* 3 for video streaming. We observe that the bandwidth of *Flow* 3 obtained by Without-FC gets reduced after $t = 26$ seconds due to the congestion on link 7→3. Hence, the quality of video streaming will be degraded significantly, which can be verified by the results on the luminance component's peak signal-to-noise ratio (Y-PSNR) of received video in Fig. 13. Specifically, the congestion after $t = 26$ seconds makes the corresponding Y-PSNR results very low. On the other hand, because F-FC can arrange the routing paths in a better way, the bandwidth of *Flow* 3 will not decrease after $t = 26$ seconds. The Y-PSNR results in Fig. 13 also indicate that the proposed F-FC scheme does not cause any adverse effect on the video streaming.

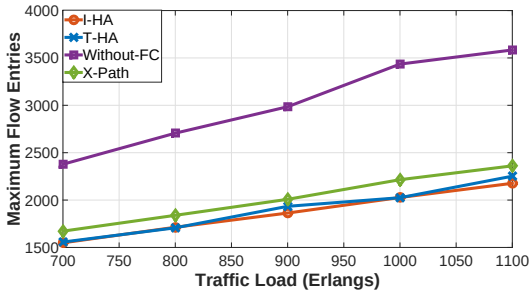
C. Dynamic Experiments for F-FC

In this experiment, we built the experimental topology based on the 14-node topology in Fig. 3. We connect a host to each POF switch to send/receive flows. Similar to the dynamic simulations described in Subsection VI-B, we design and implement two experimental scenarios to demonstrate the performance of F-FC, which are flow entry constrained and bandwidth constrained ones.

For the flow entry constrained scenario, we modify the configure file of POF switches to change their capability of storing flow entries. All the switches are equipped with Gigabit Ethernet cards, and hence the bandwidth resources are sufficient. The experimental results on total revenue and flow



(a) Total revenue.



(b) Maximum number of installed flow entries.

Fig. 15. Results for dynamic experiments with limited bandwidth.

acceptance ratio are shown in Figs. 14(a) and 14(b), respectively. We can see that the results follow the similar trends as those obtained by the simulations in Subsection VI-B. Basically, I-HA still obtains the highest revenue and flow acceptance ratio. The performance of T-HA is slightly worse than that of I-HA, but it outperforms the two benchmarks. Note that, compared with the simulation results, the performance of X-Path in this experiment looks closer to that of I-HA. This is because the experiment uses a much smaller topology and thus X-Path can converge more flows together.

For the bandwidth constrained scenario, we limit the bandwidth on each link to 90 Mbps. As shown in Fig. 15(a), the revenues from I-HA and Without-FC are similar and higher than those from the other algorithms, which confirm that our proposed F-FC scheme can achieve fine-grained TE again. The experimental results on *Flow_Mod* messages also follow the similar trends as those from the simulations, and we do not show them here for saving space. In addition, we record the maximum number of installed flow entries on the switches in Fig. 15(b), which suggest that I-HA and T-HA always require much less flow entries than the benchmarks.

VIII. Conclusion

This paper proposed a novel POF-based F-FC scheme to realize SDN-based fine-grained TE and utilized the scheme to perform a systematic case study on the forwarding plane programmability of POF. Specifically, we designed both the network system and the F-FC algorithms running on it and conducted experiments to demonstrate that our proposed scheme could not only reduce the volume of installed flow entries in switches, but also realize fine-grained TE to achieve better utilization on network bandwidth.

Acknowledgments

This work was supported in part by NSFC Project 61371117, Fundamental Research Funds for the Central Universities (WK2100060010), Natural Science Research Project for Universities in Anhui (KJ2014ZD38), and SPR Program of the CAS (XDA06010302).

References

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

[2] S. Li *et al.*, "Flexible traffic engineering (F-TE): When OpenFlow meets multi-protocol IP-forwarding," *IEEE Commun. Lett.*, vol. 18, pp. 1699–1702, Oct. 2014.

[3] OpenFlow Switch Specifications. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[4] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[5] M. Kuźniar, P. Perešíni, and D. Kostić, "What you need to know about SDN flow tables," in *Proc. of PAM 2015*, pp. 347–359. Springer, Mar. 2015.

[6] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, pp. 14–76, 2015.

[7] B. Zhang, J. Bi, and J. Wu, "AFEC: A method of aggregating forwarding equivalence classes based on overlapped paths," in *Proc. of ICNP 2012*, pp. 1–2, Nov. 2012.

[8] P. Ashwood-Smith, M. Soliman, and T. Wan, "SDN state reduction (IEFT draft)," Jul. 2013. [Online]. Available: https://tools.ietf.org/pdf/draft-ashwood-sdnrg-state-reduction-00.pdf

[9] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proc. of ACM HotSDN 2013*, pp. 127–132, Aug. 2013.

[10] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw., in Press*, 2016.

[11] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[12] Protocol independent forwarding. [Online]. Available: https://www.opennetworking.org/protocol-independent-forwarding

[13] S. Dwarakanathan, L. Bass, and L. Zhu, "Cloud application HA using SDN to ensure QoS," in *Proc. of IEEE CLOUD 2015*, pp. 1003–1007, Jun. 2015.

[14] S. Ma *et al.*, "QoS-aware flexible traffic engineering with OpenFlow-assisted agile IP-forwarding interchanging," in *Proc. of ICC 2015*, pp. 8490–8495, Jun. 2013.

[15] Z. Zhu, S. Li, and X. Chen, "Design QoS-aware multi-path provisioning strategies for efficient cloud-assisted SVC video streaming to heterogeneous clients," *IEEE Trans. Multimedia*, vol. 15, pp. 758–768, Jun. 2013.

[16] N. Xue *et al.*, "Demonstration of OpenFlow-controlled network orchestration for adaptive SVC video manycast," *IEEE Trans. Multimedia*, vol. 17, pp. 1617–1629, Sept. 2015.

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

[18] D. Lee, P. Hong, and J. Li, "RPA-RA: A resource preference aware routing algorithm in software defined network," in *Proc. of GLOBECOM 2015*, pp. 1–6, Dec. 2015.

[19] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *Proc. of INFOCOM 2013*, pp. 545–549, Apr. 2013.

[20] F. Giroire, F. Havet, and J. Moulierac, "Compressing two-dimensional routing tables with order," INRIA, Research Report RR-8658, Dec. 2014.

[21] S. Hu *et al.*, "Explicit path control in commodity data centers: Design and applications," *IEEE/ACM Trans. Netw., in Press*, pp. 1–14, 2016.

[22] M. Soliman, B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Source routed forwarding with software defined control, considerations and implications," in *Proc. of CoNEXT 2012*, pp. 43–44, Dec. 2012.

[23] S. Jyothi, M. Dong, and P. Godfrey, "Towards a flexible data center fabric with source routing," in *Proc. of SOSR 2015*, pp. 10:1–10:8, Jun. 2015.

[24] P. Filsfils, M. Previdi *et al.*, "Segment routing architecture," May 2015. [Online]. Available: https://tools.ietf.org/pdf/draft-ietf-spring-segment-routing-00.pdf
[25] L. Davoli *et al.*, "Traffic engineering with segment routing: SDN-based architectural design and open source implementation," in *Proc. of EWSDN 2015*, pp. 111–112, Sept. 2015.
[26] D. Hu *et al.*, "Design and demonstration of SDN-based flexible flow converging with protocol-oblivious forwarding (POF)," in *Proc. of GLOBECOM 2015*, pp. 1–6, Dec. 2015.
[27] S. Li, D. Hu, W. Fang, and Z. Zhu, "Source routing with protocol-oblivious forwarding (POF) to enable efficient e-health data transfers," in *Proc. of ICC 2016*, pp. 1–6, May 2016.
[28] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar, "Approximation algorithms for the unsplittable flow problem," *Algorithmica*, vol. 47, no. 1, pp. 53–78, 2007.
[29] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.
[30] J. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, pp. 712–716, Jul. 1971.
[31] H. Salama, "Multicast routing for real-time communication of high-speed networks," Ph.D. dissertation, 1996.
[32] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. of INFOCOM 2015*, pp. 657–665, Apr. 2015.