P4-TINS: P4-Driven Traffic Isolation for Network Slicing With Bandwidth Guarantee and Management

Yan-Wei Chen, Chi-Yu Li, *Member, IEEE*, Chien-Chao Tseng, and Min-Zhi Hu

***Abstract*—Network slicing is an essential technology for 5G mobile networks. It partitions network resource logically into multiple isolated slices, each of which can satisfy a suite of network requirements for one specific service. However, it cannot be fulfilled by the current SDN (Software-Defined Networks), since the conventional SDN data-plane technology, OpenFlow, is not flexible enough to offer fine-grained network resource control or queue/packet scheduling. It leads to many research studies developing corresponding solutions on programmable switches. In this work, we focus on the support of the bandwidth guarantee and management for network slices. Although several studies with the similar goal have been proposed, they do not consider interference among different flow types or use the built-in meter for easy deployment on COTS (Commercial Off-The-Shelf) P4 switches. To this end, we first conduct a case study to examine the interference cases. We then propose a solution, designated as P4-TINS (P4-driven Traffic Isolation for Network Slicing), to resolve the interference by isolating different types of traffic flows in priority queues and set the P4 switch's bucket size based on the time granularity of its bandwidth management operation. It cannot only ensure the guaranteed bandwidth for each slice but also enable coexistent slices to fairly share residual bandwidth. We have confirmed its effectiveness experimentally based on our prototype over an ONOS (Open Network Operating System) controller and a COTS P4 switch.**

***Index Terms*—P4, network slicing, programmable switch, bandwidth management.**

I. Introduction

NETWORK slicing has been a key technology for upcoming 5G mobile networks to support various services, such as eMBB (enhanced Mobile Broadband), mMTC (massive Machine Type Communications), and URLLC (Ultra Reliable Low Latency Communications). To satisfy their different network resource requirements [1], network slicing can partition network resource logically into multiple slices, each of which serves one type of service, over a shared network infrastructure [2]. The slices are isolated from each other; to avoid interference among the slices, each slice can be assigned guaranteed and maximum resource amounts, e.g., the amounts of network bandwidth. In addition to the network resource, those services may also have other QoS (Quality of Service) requirements including packet loss, packet delay, etc. These packet-level QoS requirements can be supported by queue scheduling and management solutions [3]–[10].

Thanks to the technology of the SDN (Software-Defined Networks) with a global network view and the programmable control plane, which can install data-plane flow rules dynamically on demand, the network bandwidth resource can be controlled over end-to-end network paths for network slices to some extent [11]–[14]. However, the conventional SDN data-plane technology, OpenFlow [15], is not flexible enough to offer fine-grained network resource control or queue/packet scheduling on a per-slice basis [16]–[18]. Specifically, the OpenFlow meter supports only two network actions, namely dropping packets and DSCP (Differentiated Services Code Point) remarking, with limited header formats and pipeline stages. To offer the flexibility on the data plane, there have been many research studies [3]–[7], [19] about programmable switches, and then the P4 (Programming Protocol-independent Packet Processors) switch [20] was developed and standardized [21].

The programmable P4 switch can be employed to carry out network slices, since it can dynamically execute guaranteed and maximum bandwidth on traffic flows using its core component, meter. It can run multiple meters concurrently in a pipeline; each meter can be configured to serve a set of traffic flows and apply a given bandwidth policy to it. Therefore, for each network slice, a meter can be assigned to guarantee and limit its bandwidth. Seemingly, it is straightforward; however, the default P4 switch operation may not well support the bandwidth guarantee and management on a per-slice basis. When the P4 switch classifies packets based on the concept of DiffServ with the trTCM (two rate Three Color Marker) approach [22] and then priorities guaranteed traffic using a priority queue [23], the beyond-guaranteed, yet below-maximum, traffic amounts of all the meters are sent to another low-priority queue; different conditions between high-priority and low-priority queues may cause interference between traffic flows, thereby impeding bandwidth guarantee and management.

We thus conduct a case study to examine the effectiveness of the bandwidth guarantee and management on a COTS P4 switch with the Intel/Barefoot Tofino P4 chip. We use a meter to serve each slice with given guaranteed and maximum

Manuscript received 2 September 2021; revised 9 January 2022 and 3 March 2022; accepted 8 March 2022. Date of publication 14 March 2022; date of current version 12 October 2022. This work was supported in part by the Center for Open Intelligent Connectivity from the Featured Areas Research Center Program within the Framework of the Higher Education Sprout Project by the Ministry of Education, Taiwan; in part by the Ministry of Science and Technology, Taiwan, under Grant 109-2628-E-009-001-MY3, Grant 110-2221-E-A49-031-MY3, Grant 110-2221-E-A49-044-MY3, Grant 110-2221-E-A49-064-MY3, Grant 110-2224-E-011-002, and Grant 110-2224-E-A49-002; in part by the Ministry of Economic Affairs, Taiwan, under Grant 107-EC-17-A-02-S5-007; and in part by the National Defense Science and Technology Academic Collaborative Research Project in 2022. The associate editor coordinating the review of this article and approving it for publication was A. Detti. *(Corresponding author: Chi-Yu Li.)*

The authors are with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ghj0504520.cs07g@nctu.edu.tw; chiyuli@cs.nctu.edu.tw; cctseng@g2.nctu.edu.tw; cry589036511.cs09@nycu.edu.tw).

Digital Object Identifier 10.1109/TNSM.2022.3159232

bandwidth. There are two major findings. First, interference may happen between different flow types so that the residual bandwidth, which is the amount left by the aggregate guaranteed bandwidth, may be shared unfairly among different flows. There are three major interference cases: non-sliced and sliced TCP flows, sliced TCP and UDP flows, and non-sliced mice and elephant flows. Second, the P4 switch's bucket size shall be set based on the time granularity of its bandwidth management operation; otherwise, the guaranteed bandwidth of sliced TCP flows may not be ensured.

Although some studies [23]–[26] have been proposed for network slicing on the P4 switch, they do not consider interference among different flow types or use the built-in meter for easy deployment on COTS P4 switches. Specifically, the third study [25] proposed to use virtual queues for traffic management in the P4 pipeline to meet latency requirements of network slices, but it cannot achieve bandwidth guarantee or resolve flow interference. The last study [26] designed a P4 pipeline solution for bandwidth management on TCP flows, in order to resolve poor TCP performance from the P4 meter. However, we find that the root cause lies in the bucket size configured in the P4 switch, but is not a lack of the TCP-specific bandwidth management solution; it has been experimentally validated in this work.

We then propose an effective, interference-free solution, designated as P4-TINS (P4-driven Traffic Isolation for Network Slicing), for the bandwidth guarantee and management on network slices. P4-TINS adopts a two-level priority queue framework where a meter serving each slice receives all the traffic belonging to the slice and dispatches packets to high and low priority queues. To resolve the interference, the major idea is to isolate traffic in low priority queues based on different traffic types. In the meter pipeline, we develop slice identification, packet classification, elephant flow identification, and policer modules to achieve the goal. The solution is simple and compliant to the P4 standard [21] so that it can be easily deployed on the COTS P4 switches.

We prototype P4-TINS using an ONOS (Open Network Operating System) controller [27] with built-in applications and two newly developed applications, namely bandwidth management and elephant-mice separation, on the programmable P4 switch. We evaluate it in both single-slice and multi-slice cases. The evaluation result shows that in the single-slice case, P4-TINS can prevent a sliced TCP flow from being interfered by non-sliced UDP/TCP flows so that the sliced flow can receive more residual bandwidth up to 25.50 times. In the multi-slice case with multiple flows, besides that each slice's guaranteed bandwidth can be ensured, the slices with intensive traffic (e.g., four or more flows in a slice) can fairly share residual bandwidth with only up to 4% difference, and the flows in each slice can fairly share the slice's bandwidth with only small differences ranging from 0.54% to 8.23%. For the multi-slice case with many flows, P4-TINS can always ensure each slice's guaranteed bandwidth and allow coexistent slices to share residual bandwidth without interference.

The rest of this paper is structured as follows. Section II introduces the background of the P4 switch and the related work. In Section III, we conduct an experimental case study on the COTS P4 switch. We then design and implement the solution, P4-TINS, in Section IV. Sections V, VI and VII evaluate P4-TINS, discuss issues and conclude the paper, respectively.

II. Background and Related Work

In this section, we first introduce the major operation of the P4 switch and then present the related work.

A. P4 Switch

The P4 switch has a critical component, meter, for executing quality of service (QoS) on traffic flows. The meter can classify traffic flows and collect traffic statistics at run time, and then apply different policies, which are executed on a per-packet basis, to classified traffic. The P4 meter runs the trTCM algorithm [22] by default. The trTCM monitors the data rates of traffic flows and classifies their packets based on two configured parameters, namely Peak Information Rate (PIR) and Committed Information Rate (CIR). They are the maximum bit rate and the guaranteed bit rate, respectively, for the meter. The trTCM maintains two token buckets, P and C, with rates PIR and CIR, respectively. The maximum size of the token bucket P is Peak Burst Size (PBS) and that of the token bucket C is Committed Burst Size (CBS). Initially, the token buckets P and C are full; thereafter, their token counts are incremented by one PIR times per second up to PBS and one CIR times per second up to CBS, respectively. Each incoming packet consumes a token and is classified into three colors, namely green, yellow, and red based on the following three conditions, respectively: (1) both the buckets P and C still have tokens; (2) the bucket P has tokens, but the bucket C does not have any token; (3) neither of the buckets P and C have tokens. Then, the policer can give different action policies of packet processing to different colors.

B. Related Work

There have been many research studies with a similar goal of this study, bandwidth guarantee/management, and about programmable switches. In the following, we first present control-plane solutions with the similar goal and then discuss data-plane solutions with different performance goals, especially for bandwidth guarantee/management, on programmable switches.

Several SDN studies [12], [14], [16]–[18] have been proposed to achieve bandwidth guarantee and management by leveraging the control-plane programmability from SDN. For example, Moeyersons *et al.* [14] proposed a solution in the SDN network to guarantee bandwidth for emergency flows while maximizing the total rate of best effort flows. The solution manipulates routing paths of traffic flows and regulates the bandwidth guarantee of each emergency flow along its route from the control plane. Morin *et al.* [12] introduced a QoS control mechanism on the SDN controller using on-demand MPLS (Multi-protocol Label Switching) tunnels with guaranteed bandwidths. Traffic flows are classified into multiple MPLS tunnels, and each tunnel's packets are labeled with a priority. The QoS control module can then enforce a QoS policy for each priority level into the data plane. However, the conventional SDN data plane based on OpenFlow [15] switches with fixed functionality lacks for flexibility, e.g.,

only two supported metering actions, namely dropping packets and DSCP remarking, and limited header formats. To offer the flexibility on the data plane, some research studies on programmable switches were proposed and then the P4 standard [21] was introduced for the development of P4-based programmable switches. The present study focuses on the P4-based programmable switches for achieving bandwidth guarantee and management, instead of the control plane, but can be used to facilitate the SDN networks with P4-based data planes.

Other than the goal of bandwidth guarantee and management, there have been many studies about programmable switches in terms of other different goals, such as programmable queues/schedulers for packet scheduling [3]–[7], [19] and meeting latency requirements of network slices with bandwidth management [25]. Differently, the present study focuses on achieving per-slice bandwidth guarantee and management solution, which the above studies do not cover.

Several other studies [23], [24], [26] on programmable switches have been proposed for the P4 switch in terms of the goal of bandwidth guarantee and management. Specifically, Chen *et al.* [23] adopted two-level priority queue for the trTCM on the P4 switch to carry out slice-based bandwidth limit and guarantee, but they considered only UDP traffic but not TCP traffic or the interference between them. Tokmakov *et al.* [24] introduced a traffic management algorithm that combines rate-limited strict priority and deficit round-robin for latency-aware and fair scheduling in the data center using a P4 software switch. Wang *et al.* [26] designed a P4 pipeline solution for bandwidth management on TCP flows, in order to resolve poor TCP performance from the P4 meter. However, these two studies do not cover bandwidth guarantee or residual bandwidth allocation for network slices. Moreover, the authors [26] discovered that the interaction between the meter and the TCP congestion control prevents TCP flows from reaching expected throughput so that they proposed a TCP-specific bandwidth management solution, but we found that configuring the bucket size based on time granularity of the bandwidth management operation can address the issue.

Different from the above studies, the present study not only studies the impact of the traffic interference, which may impede fair allocation of residual bandwidth, and the impact of bucket size, which may prevent TCP slices from achieving guaranteed bandwidth, but also proposes/implements the P4-TINS solution to resolve the interference and evaluates it on a real P4 platform.

III. Case Study

In this section, we conduct a case study to examine the effectiveness of bandwidth management for traffic slicing on a COTS P4 switch. We first consider only UDP or TCP flows in slices, and then examine the impact of traffic contention between UDP/TCP flows and that between mice/elephant flows.

For the default P4-based bandwidth management, we employ a common solution where packets in a slice are classified based on the concept of DiffServ with the trTCM approach [22] and then prioritized by a priority queue [23].

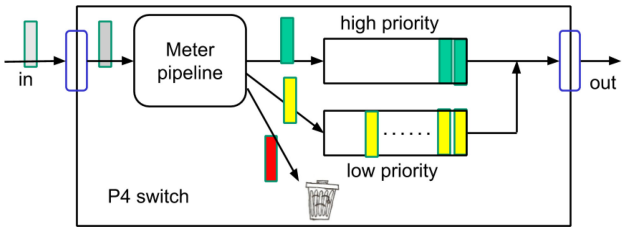


Fig. 1. A common solution with the trTCM approach and a priority queue for the P4-based bandwidth management.

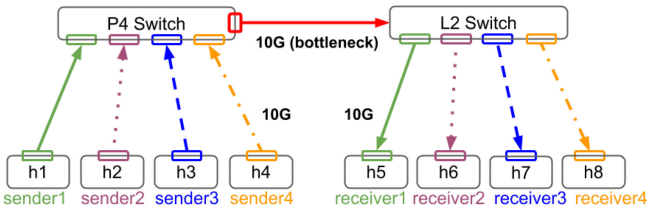


Fig. 2. An experimental P4 testbed used in this work.

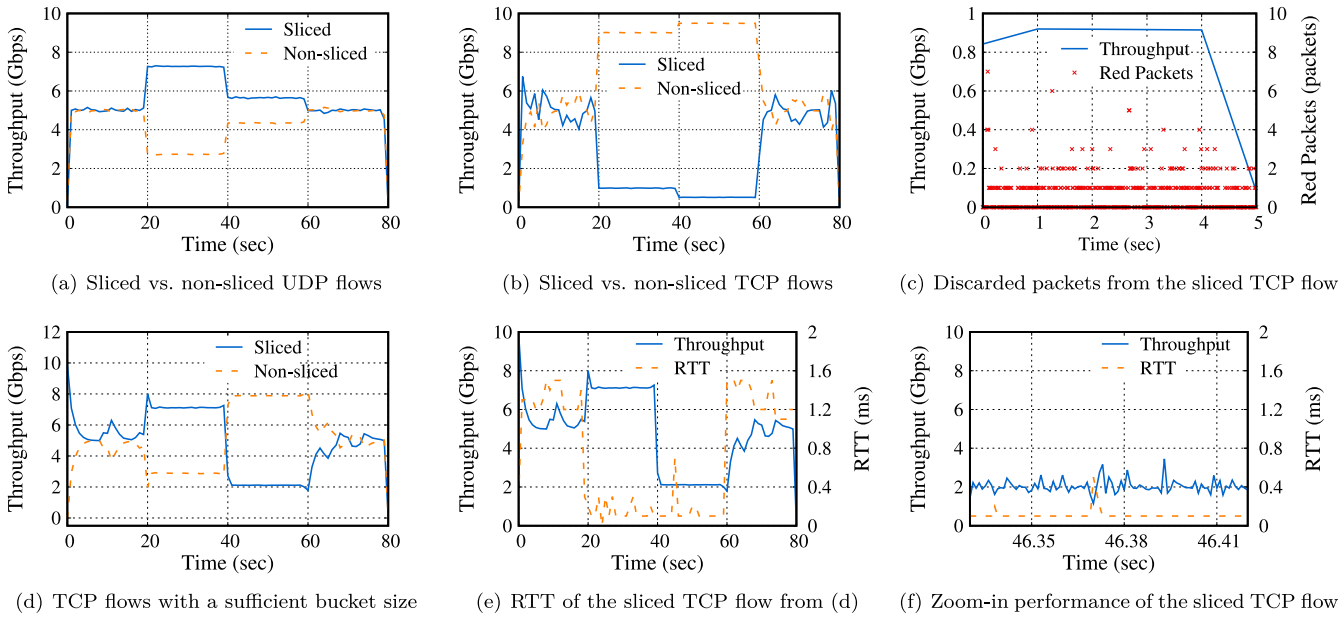
As shown in Figure 1, sliced packets are classified into green, yellow, and red packets by a meter; the green and yellow packets are dispatched to high-priority and low-priority queues, respectively, whereas the red ones are discarded. The bandwidth management is achieved by setting the bandwidth of green packets for the target slice's guaranteed bandwidth, i.e., CIR, and setting the total bandwidth of green and yellow packets for its limited bandwidth, i.e., PIR. Note that the packets of non-sliced flows are all sent to the low-priority queue.

A. Experimental Platform and Setting

Figure 2 shows the P4 testbed for this case study. The P4 switch is the Inventec D10056 programmable switch with an Intel/Barefoot Tofino P4 chip. The chip with the protocol-independent switch architecture (PISA) can be programmed using the P4-16 language [28]. It can support up to 8 queues and thus 8 priority levels. The aforementioned P4-based bandwidth management is enabled on this P4 switch. It connects to another layer-2 switch, which is used to gauge the throughput of each traffic flow coming out from the P4 switch. Each of these two switches connects to four servers, the model of which is QCT D51B-1U. Each server has two Intel Xeon E5-2630 CPUs at 3.1 GHz and an Intel X710 10GbE NIC. It runs Ubuntu 16.04 with Linux kernel v4.15.0 and 1500-byte MTU. The bandwidth of each link is 10 Gbps. In each experiment, traffic flows are sent from the servers connecting the P4 switch to those connecting the other switch. We use iPerf to generate UDP/TCP flows and adopt Flowgrind [29] to measure RTT on TCP flows. Notably, TCP New Reno is used for TCP flows in this work.

To test the effectiveness of the default bandwidth management on traffic slicing, we use the following setting unless explicitly specified. Each test runs for 80 seconds. During the first 19 seconds, no slice is enabled. For the next 20 seconds from 20th to 39th second,[1] a slice is enabled for sliced traffic flows with a configuration $S(7, 8)$, which represents that

[1]We represent a time interval from a-th to b-th second as I[a, b] hereafter. For example, the interval between 20th and 39th seconds is represented as I[20, 39].



(a) Sliced vs. non-sliced UDP flows (b) Sliced vs. non-sliced TCP flows (c) Discarded packets from the sliced TCP flow

(d) TCP flows with a sufficient bucket size (e) RTT of the sliced TCP flow from (d) (f) Zoom-in performance of the sliced TCP flow

Fig. 3. The effectiveness of the default bandwidth management is examined on sliced UDP and TCP flows.

the slice S has 7 Gbps CIR and 8 Gbps PIR. The configuration is then changed to $S(2, 10)$ for the next 20 seconds as I[40, 59]. Afterwards, the P4 switch reverts to the state without any slice. Note that each test has 10 runs. When the 10-run results are observed to have the same trend, a representative trace is chosen to present the performance change over time. The performance is measured every second, and the presented performance values come from the average over a test interval, 20 seconds. For bar graphs, upper and lower bounds of performance values are also plotted.

B. Sliced UDP Flow

We first examine the effectiveness of the default bandwidth management on a sliced UDP flow by considering a case that it coexists with a non-sliced UDP flow. Both of the UDP flows are generated with 10 Gbps to pass through the P4 switch. Figure 3(a) shows that the sliced UDP flow can always obtain the throughput larger than CIR but smaller than PIR. The reason is that the CIR bandwidth is guaranteed by the high priority queue and the amount of the sliced traffic flowing into the low priority queue is constrained by the difference between PIR and CIR.

The sliced UDP flow can achieve 7.27 Gbps and 5.64 Gbps on average during the intervals I[20, 39] and I[40, 59], respectively. Take the former slicing period with $S(7, 8)$ as an example. The throughput 7.27 Gbps is larger than 7 Mbps CIR, and the difference 0.27 Gbps comes from the low priority queue, where 1 Gbps from the sliced UDP flow contends with 10 Gbps from the non-sliced UDP flow. From the contention, the sliced and non-sliced UDP flows obtain 0.27 Gbps and 2.72 Gbps, respectively; the beyond-guaranteed bandwidth of the sliced flow and the total bandwidth of the non-sliced one approximately have the similar ratio of their traffic amounts going to the low-priority queue, i.e., 0.1.

We do observe the same trend from the latter slicing period with $S(2, 10)$. The sliced and non-slice UDP flows get 3.64 Gbps and 4.35 Gbps, respectively, from the contention of the low priority queue with 8 Gbps ($10 - 2 = 8$) and 10 Gbps. The result shows that the sliced UDP flow can receive residual bandwidth, which represents the amount left by guaranteed bandwidth, in proportion to the ratio of non-guaranteed, allowable bandwidth (PIR - CIR) from the sliced flow to the traffic volume of the non-sliced flow.

C. Sliced TCP Flow

We next examine the effectiveness on a sliced TCP flow by considering the coexistence of the flow and another non-sliced TCP flow. The maximum bandwidth of the TCP flows is set to 10 Gbps. Figure 3(b) shows the throughput of these two TCP flows over time. It is observed that the sliced TCP flow obtains only 985.77 Mbps and 511.48 Mbps on average during the intervals I[20, 39] and I[40, 59], respectively. They are much lower than the guaranteed bandwidth values, 7 Gbps and 2 Gbps, respectively.

We further examine the performance issue by checking the patterns of discarded red packets for the interval I[36, 40], as shown in Figure 3(c). It is observed that given the guaranteed bandwidth as high as 7 Gbps, there are still many discarded packets, which are classified as red packets in the trTCM approach, with only throughput below 1 Gbps. The discarded packets prevent the congestion window of the sliced TCP flow from growing so that its throughput is limited.

We discover that the discard can be attributed to a small bucket size of the meter, which controls the allowable burst traffic volume of the flow. By default, the bucket size is 5K bytes. To validate the impact of the bucket size, we vary it in terms of transmitted bytes from 10 Gbps traffic within 0.01 ms, 0.1 ms, 1 ms, 10 ms, and 100 ms. As shown in Figure 4, these five cases of the bucket size are conducted from 5th

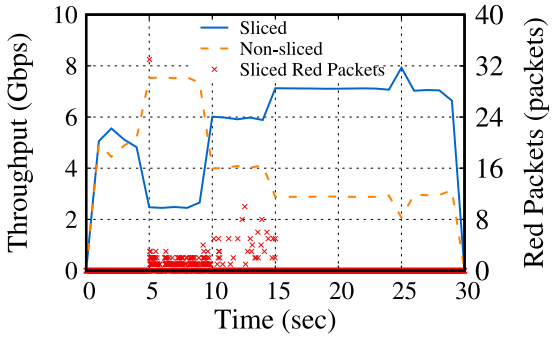
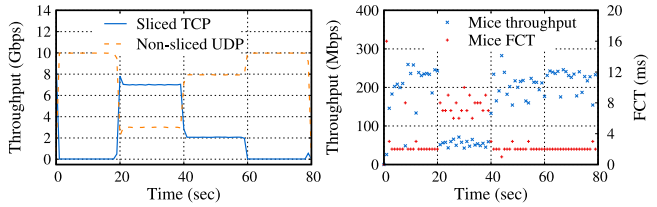


Fig. 4. Impact on discarded packets from various bucket sizes: transmitted bytes from 10 Gbps traffic within 0.01/0.1/1/10/100 ms; these five cases are conducted in the time interval I[6, 30] in sequence with 5 seconds each.



(a) UDP/TCP Flows (b) Mice/Elephant Flows

Fig. 5. The impact of the default bandwidth management on the contention of different flow types.

second to 30th second in sequence with 5 seconds each. It is observed that no packets are dropped when the bucket size is not smaller than the amount of bytes generated by 10 Gbps traffic within 1 ms (i.e., 1.25 MB). It can be inferred that the meter's bandwidth management operates at a time granularity that is not greater than 1 ms.

We conduct this experiment again with a sufficient bucket size, 1.25 MB. Figure 3(d) shows that the sliced TCP flow can achieve guaranteed bandwidth values, 7 Gbps and 2 Gbps, during the intervals I[20, 39] and I[40, 59], respectively. Specifically, the average throughput results are only 7.17 Gbps and 2.14 Gbps, respectively. They are approximately equal to CIR plus a small constant. Thus, the bucket size of a P4 switch shall be determined based on the time granularity of its bandwidth management operation.

However, it cannot receive residual bandwidth as the sliced UDP flow in proportion to the ratio of its non-guaranteed, allowable bandwidth to the other flow's traffic volume. The root cause is that there is a mismatch between the operations of the high-priority and low-priority queues for a sliced TCP flow; the mismatch can prevent a TCP congestion window from growing. When a TCP flow is sliced into the high-priority queue, its RTT can be greatly shorten, as shown in Figure 3(e). The average RTT value of the TCP flow decreases from 1.22 ms in the first 20 seconds without any slice to 0.15 ms with a slice $S(7, 8)$. However, the packets put into the low-priority queue cannot sustain such low RTT, since they are congested with the packets of the non-sliced TCP flow. These packets can cause packet timeout to occur. Figure 3(f) shows a zoom-in view of the throughput and RTT performance for the slice $S(2, 10)$; it can be observed that the throughput fluctuates over time frequently.

Moreover, the congestion in the low-priority queue can cause out-of-order packets of the sliced TCP flow. We observe that the queue length of the high-priority queue is only 25 packets on average, but that of the low-priority queue can be up to almost 15 thousand packets. Some packets of the sliced TCP flow are scattered over the low-priority queue among that large amount of other non-sliced packets, so they may not only cause timeout but also experience out-of-order delivery.

In sum, the non-sliced TCP packets in the low-priority queue can hinder the sliced TCP flow from contending for residual bandwidth. So, *non-sliced and sliced TCP flows shall not share the same low-priority queue.*

D. UDP/TCP Flows Contention

We next examine the impact of bandwidth management on the contention between UDP and TCP flows. We generate one sliced TCP flow with 10 Gbps maximum throughput and one non-sliced 10 Gbps UDP flow, with a sufficient bucket size. Figure 5(a) shows the average throughput of those two flows over time. We observe the similar issue that the sliced TCP flow obtains only limited residual bandwidth; specifically, it gets only average throughput 7.05 Gbps and 2.10 Gbps with slice settings $S(7, 8)$ and $S(2, 10)$, respectively. The non-sliced UDP flow grabs almost all the residual bandwidth.

To analyze the root cause, we further examine the change of queue length over time. The queue length of the high-priority queue is still short with only 25 packets on average, whereas that of the low-priority queue can be as large as about 24 thousand packets. It is because the UDP flow keeps pushing packets into the low-priority queue without any congestion control. Such congestion makes the TCP packets in the low-priority queue cause timeout and out-of-order delivery to the TCP flow. The very limited residual bandwidth sharing can be attributed to the aforementioned operation mismatch issue; the amount of the residual bandwidth becomes much smaller than that in Section III-C because of the aggressive UDP flow. Therefore, *the sliced TCP and UDP flows shall not either share the same low-priority queue.*

E. Mice/Elephant Flows Contention

We next examine the impact of bandwidth management on the contention of mice and elephant TCP flows. We generate a mice TCP flow using back-to-back HTTP requests/responses during the whole 80s experiment period. Two elephant TCP flows with 10 Gbps each are generated within the time interval I[20, 59]. A slice, $S(7, 10)$, is enabled for all the elephant and mice flows during the last 20 seconds, i.e., I[40, 59].

Figure 5(b) shows the throughput and flow completion time (FCT) of the mice flow over time. It is observed that when those two elephant flows are generated without any slice, the throughput and FCT of the mice flow degrade to 54.86 Mbps and increase to 7.58 ms, respectively. Once the mice flow is sliced with the elephant flows, the throughput and FCT values return to the normal result, which gives 204.53 Mbps and 2.05 ms, respectively, with the mice flow only. It shows that

the slicing can prevent the mice flow's performance from being affected by the other elephant flows, even if they stay in the same slice. Notably, the aggregate throughput of elephant and mice flows still achieves next to 10 Gbps for all the time, though only the throughput of the mice flow is shown in the figure.

The root cause is that when there is not any slice, all the flows put packets into the same queue and thus cause a large amount of queued packets, which congest the queue and may hurt the mice flow's performance. When the mice flow is inside the slice, its packets in small amount can be put into the high-priority queue, which is not congested, because of flow fairness on the packet scheduling. The mice flow can receive the same performance as the case with the mice flow only, Thus, non-sliced mice flows shall not share the same queue with non-sliced elephant flows, whereas sliced mice flows can share the same slice with the sliced elephant ones.

F. Summary

We observe three major cases of the interference between different flow types: non-sliced/sliced TCP flows, UDP and sliced TCP flows, and non-sliced mice/elephant flows. Their interference conditions all happen in low-priority queues. For sliced TCP flows without receiving proportional residual bandwidth, there is a mismatched operation between high-priority and low-priority queues due to much more congested low-priority queues. It can be caused by UDP flows and non-sliced TCP flows. The non-sliced TCP flows increase congestion windows purely based on the condition of low-priority queues but are not constrained by a PIR as the sliced TCP flows. From the UDP flows without congestion control, the packets with an amount as the same as the difference between CIR and PIR are always sent to low-priority queues. Both of them may generate an excessive amount of packets which the bandwidths of low-priority queues cannot accommodate. Therefore, the packets of the sliced TCP flows in the low-priority queue may suffer from out-of-order delivery and timeout.

Moreover, the packets of non-sliced mice flows can be affected by the congestion, which may be caused by non-sliced/sliced elephant flows, in the low-priority queue. Once mice flows are sliced, small queue length of the high-priority queue can protect them from being thwarted by other flows; the small queue length can be kept since only the green packets with guaranteed bandwidth are dispatched to the high-priority queue and can be sent out immediately. Another lesson learned is that the P4 switch's bucket size shall be set based on the time granularity of its bandwidth management operation. Given a link capacity C and a time granularity T at which the buckets are replenished, the bucket size B shall be set to satisfy $B \geq C \times T$. For example, $T \leq 1$ *ms* is observed for the used P4 switch, so with $C = 10$ *Gbps*, B shall be set to at least 1.25 MB.

Note that although the default operation can satisfy the guaranteed and maximum bit rates, it cannot well support the management of residual bandwidth. With the existence of interference, the sliced TCP flows receive only little residual bandwidth while coexisting with non-sliced UDP/TCP flows, which can receive most of it. It may not make sense, since the sliced flows are usually those to be prioritized but receive only next to the guaranteed bandwidth which may be smaller than the bandwidth obtained by other coexistent non-sliced flows. Moreover, in order to manage the residual bandwidth allocation, the prerequisite is to resolve the interference; then, different weights can be effectively applied to various traffic types or low priority queues.

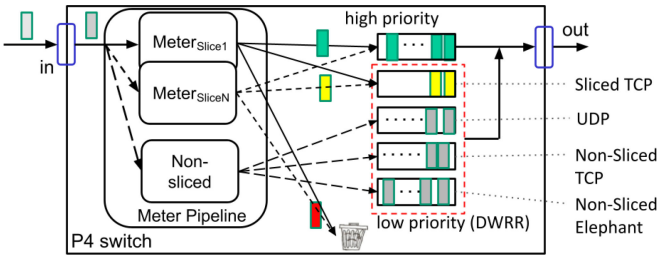


Fig. 6. The design overview of P4-TINS traffic scheduling.

IV. P4-TINS Solution

We seek to propose a network slicing solution for the per-slice bandwidth guarantee and management. It cannot only provide effective bandwidth guarantee but also allow residual bandwidth to be shared in an interference-free way. Although we can use trTCM to achieve effective bandwidth guarantee, there are interference issues between different traffic types according to the aforementioned case study. To achieve the goal, we design a solution called P4-TINS (P4-driven Traffic Isolation for Networking Slicing). The major idea is to resolve the interference by isolating traffic in low priority queues while applying trTCM. Without the interference, the residual bandwidth allocated to different traffic types can be thus customized based on weight assignment among the low-priority queues.

A. Design

P4-TINS relies on the conventional two-level priority queue scheduler with a trTCM meter pipeline, but separates low-priority queues for different types of traffic. Based on the lessons learned from Section III, three levels of traffic isolation are needed: (1) non-sliced elephant and mice flows; (2) TCP and UDP flows; (3) non-slice and sliced TCP flows. We thus create four low-priority queues as shown in Figure 6: non-sliced elephant, UDP, non-sliced TCP, and sliced TCP flows. Then, we apply deficit weighted round robin (DWRR) to these low-priority queues. Its weights are configurable to enable customizable allocation of the residual bandwidth among different traffic types.

Each slice is contained by a meter, as shown in Figure 6, and the green packets of all the slices are sent to the high priority queue. The aggregate guaranteed bandwidth of all the slices shall not exceed the link capacity (here, it is 10 Gbps), so a new slice that requests an amount of guaranteed bandwidth more than the residual bandwidth is rejected to be established. In addition, the number of supported slices is also constrained by the maximum number of meters supported by the P4 switch. It can be more than ten thousand; for example, the used P4 switch supports up to 35840 meters.

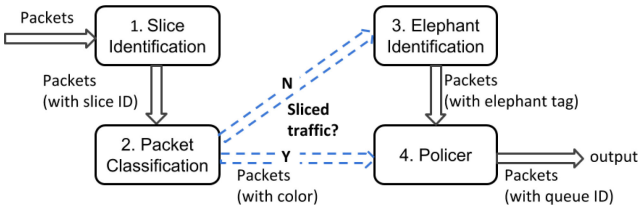


Fig. 7. P4-TINS: the design logic of meter pipeline.

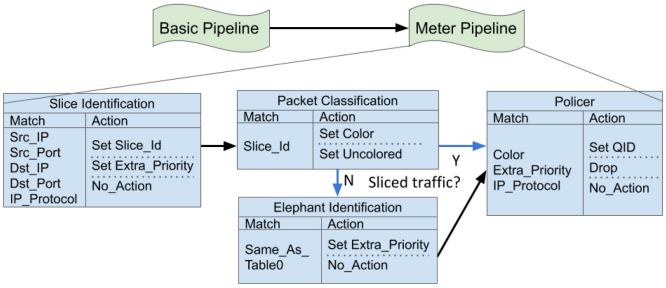


Fig. 8. P4-TINS: the flow tables of meter pipeline.

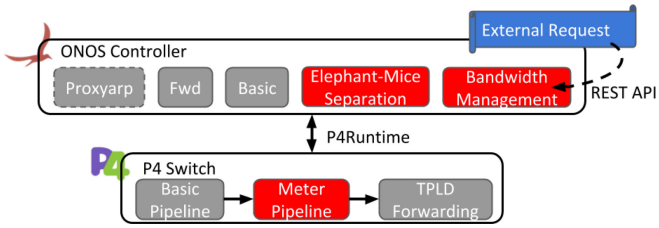


Fig. 9. P4-TINS architecture with ONOS controller and P4 switch.

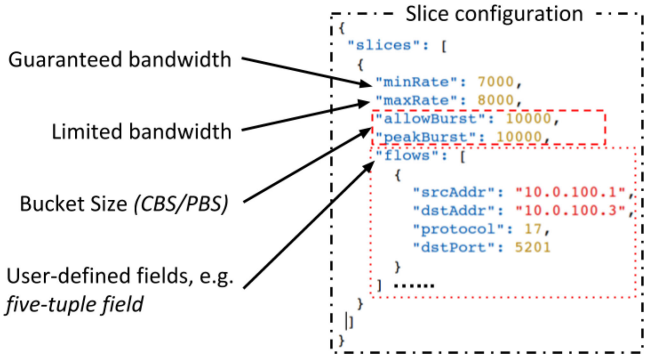


Fig. 10. P4-TINS: an example of the bandwidth management request on a traffic slice in the format of RESTful API.

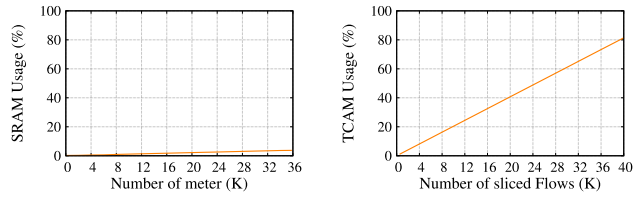
One major issue arising from the support of the low-priority queues is how to enable the meter pipeline to correctly dispatch packets to different high-priority and low-priority queues. Figure 7 shows the design logic of the P4-TINS meter pipeline. It consists of four modules: slice identification, packet classification, elephant identification, and policer; the flow table of each module is shown in Figure 8. The packets entering the meter pipeline are firstly handled by the slice identification module. This module identifies packets of sliced flows based on user-defined header fields and the slice configuration, and tags them with their slice identifiers in the per-packet metadata. The packet classification module further classifies each packet into three colors based on the trTCM method by considering the per-slice meter of the slice to which the packet belongs.

Afterwards, when the packets do not belong to any sliced traffic flows, they are forwarded to the elephant identification module; otherwise, they go to the policer. The elephant identification module detects elephant flows from non-sliced TCP flows based on whether a flow's instantaneous throughput is larger than a specified threshold, say α, and then tags each identified elephant flow. An application in the control plane is developed for the detection; it periodically queries flow counters to gauge the throughput. Once an elephant flow is identified, the application installs a rule that can tag the flow's packets in the flow table of elephant identification. Finally, the policer dispatches sliced green packets to the high-priority queue, and both sliced yellow and non-sliced packets to the low-priority queues; the red packets are dropped. For the low-priority queues, the policer differentiates packets of sliced TCP, UDP, non-sliced TCP, and elephant flows based on sliced ID, transport-layer header, and elephant tag.

B. Implementation

We prototype P4-TINS using an ONOS controller with several built-in applications and a programmable P4 switch, which serve the control and data planes, respectively. The controller and the switch use P4Runtime [30] to communicate with each other. Figure 9 shows the architecture of P4-TINS. We develop two new control-plane applications for P4-TINS: elephant-mice separation and bandwidth management. In the P4 switch equipped with a basic pipeline module developed by ONF, a meter pipeline, we carry out the packet scheduling of P4-TINS in the meter pipeline and its two-level priority queue framework (see Figure 6) as the TPLD (Two-level Priority with Low-priority DWRR) forwarding module.

Those two newly deployed applications at the ONOS controller are both implemented in Java. The bandwidth management application handles external bandwidth management requests by managing requested traffic slices and configuring trTCM parameters of slice meters. It installs flow rules into customized P4-TINS flow tables in the meter pipeline for slice identification, packet classification, and policy enforcement. It provides RESTful API for the external bandwidth management on traffic slices. Figure 10 shows an example of a slice request with a guaranteed bandwidth (minRate), a limited bandwidth (maxRate), bucket sizes (allowBurst and peakBurst), and corresponding flows. The application accepts a newly arrival request only when its requested minRate (i.e., guaranteed bandwidth) does not exceed currently available link bandwidth. According to the finding about bucket size determination in Section III, we configure the bucket size to be the number of available bytes that can be transmitted with the given link capacity 10 Gbps within 100 ms (here, it is 125 MB). The reason is that the P4 standard [21] recommends that the maximum burst size should be set as large as the number of bytes that can be transmitted across the maximum speed port within 100 ms, though the case study in Section III-C has shown that 1.25 MB is sufficient for the bucket size of the used P4 switch.



(a) Increasing number of meters/slices

(b) Increasing number of sliced flows

Fig. 11. P4-TINS overhead in the usage of SRAM and TCAM.

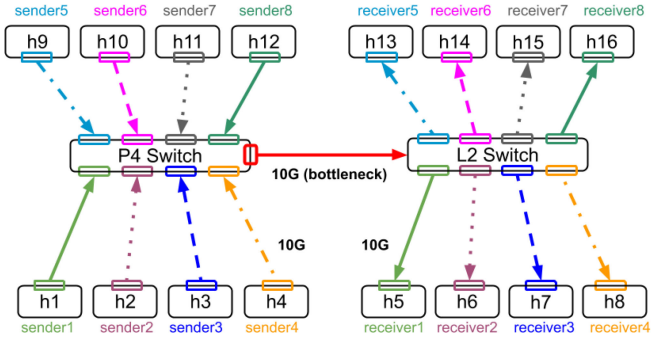


Fig. 12. An experimental P4 testbed with 8 senders and 8 receivers.

The elephant-mice separation application detects elephant TCP flows from non-sliced traffic. The detection criterion is based on whether the instantaneous throughput is larger than a specified threshold, α, or not (here, $\alpha = 300$ Mbps). It periodically queries flow counters of the forwarding table every β seconds (here, $\beta = 3$), and calculates the throughput based on the increased byte amount between two consecutive queries. Once the elephant throughput is detected for a flow with γ times (here, $\gamma = 2$), the flow is recognized as an elephant flow. Upon the detection of an elephant TCP flow, the application installs an identification rule that marks the flow's packets at the P4 switch. Note that we here aim to show that the separation of elephant and mice flows can benefit the performance of mice flows, so for the prototype, the elephant-flow detection is simply implemented on the controller with coarse-grained flow statistics based on the periodic queries. To achieve low-latency detection performance, the elephant-mice separation should be implemented on the P4 switch.

Note that for the scalability of P4-TINS, the supported number of sliced flows depends on the maximum size of the flow table, slice identification, as shown in Figure 8, since the flow table maintains slicing information about the mapping between the 5-tuple information and slice ID of each sliced flow. Moreover, the maximum table size is constrained by the size of TCAM (Ternary Content Addressable Memory). To understand the scalability, we examined the maximum table size on the P4 testbed by generating many different flows and found that it can be up to 40960 flows. It is also the maximum supported number of flows in a slice when all the flows have the same slice ID.

V. Evaluation

In this section, we evaluate P4-TINS using the prototype on the COTS P4 switch in non-slice, single-slice, and multi-slice cases, some of which are with many flows. The evaluation testbed is similar to the one in the case study, but has more senders and receivers (i.e., 8) to test multi-slice cases, as shown in Figure 12. Each test has 10 runs. The other experimental settings are the same as those in Section III-A. Notably, the aggregate CIR amounts do not exceed the maximum bandwidth 10 Gbps.

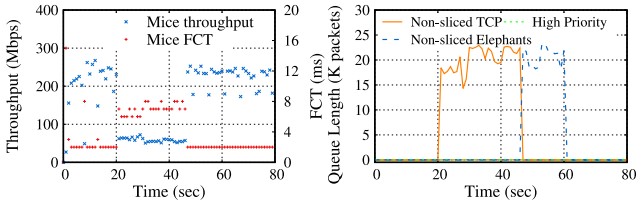
Note that the reasons why the most related state-of-the-art solutions [23], [24], [26] are not compared with P4-TINS in the evaluation are as follows. First, [24] focuses on only rate limit, so it certainly cannot achieve bandwidth guarantee or fair residual bandwidth share. Second, the case study of Section III has been conducted based on the two-level priority queue scheduler with a single low-priority queue (see Figure 1) from [23], so the results in the case study can already show that the solution [23] can suffer from the contention among UDP/TCP flows and that among elephant/mice flows. Third, [26] proposed a TCP-specific bandwidth management solution to address the poor TCP performance observed on the P4 switch; however, we discovered that the root cause lies in an improper bucket size configured in the P4 switch rather than a lack of the new solution. Moreover, Section III-C has already showed that configuring a proper bucket size based on time granularity of the bandwidth management operation can address the poor TCP performance issue.

A. P4-TINS Overhead

To examine the overhead of P4-TINS, we increase the number of meters and sliced flows while observing the usage of SRAM and TCAM resources. There are two observations. First, the maximum number of meters, each of which is used for a slice, is 35840 on the P4 platform; that is, the maximum number of supported slices can be up to that number. A meter entry is created for each meter/slice in the packet classification table, as shown in Figure 8, and it consumes only the resource of SRAM. As shown in Figure 11(a), the maximum number of meters consumes little SRAM resource without exceeding 5%. Second, the maximum number of supported sliced flows can be up to 40960. A sliced flow has a flow entry associated with its slice ID in the slice identification, and it consumes TCAM resource. Figure 11(b) shows that every 2000 flows increase about 4% usage of TCAM, and the maximum number of supported sliced flows requires more than 80% TCAM resource. Notably, register resource is not used in the implementation of P4-TINS.

B. Non-Slice Case: Mice and Elephant TCP Flows

We examine the effectiveness of P4-TINS in the case where non-sliced elephant and mice TCP flow coexist. We generate a mice TCP flow during the whole 80s experiment and two 10 Gbps elephant TCP flows only during the interval I[20, 59]; P4-TINS is enabled only during the interval I[40, 59]. Figure 13(a) shows the throughput and FCT results of the mice TCP flow over time. It is observed that without P4-TINS, the mice TCP flow is largely affected by the elephant TCP flows and can achieve only 58.90 Mbps throughput with



(a) Throughput and FCT (b) Queue length

Fig. 13. P4-TINS: performance of the non-sliced mice TCP flow.

7 ms FCT on average during I[21, 46]. After enabling P4-TINS at the 40th second, it takes around 6 seconds to take effect on the protection of the mice flow; it then obtains 230.26 Mbps throughput with 2 ms FCT on average during I[47, 80], where no TCP flows are generated in the last 20 seconds. Note that the reason why the detection takes 6 seconds is that each flow's throughput is checked every 3 seconds (i.e., $\beta = 3$) and an elephant flow is recognized whenever its elephant throughput is detected twice (i.e., $\gamma = 2$) successively. We seek to show that the separation of elephant and mice flows can benefit the performance of mice flows, so the elephant-flow detection is simply implemented on the controller with coarse-grained flow statistics. Once it is implemented on the P4 switch, the detection time can be greatly reduced. We leave it to our future work.

Figure 13(b) shows that the length of the non-sliced TCP queue is reduced to next to zero and that of the non-sliced elephant queue is largely increased after P4-TINS detects the elephant flows and dispatches their packets to the non-sliced elephant queue instead of the non-sliced TCP queue. With the negligible number of queued packets in the non-sliced TCP queue, the mice flow can be served without the interference from the elephant flows.

C. Single-Slice Case

In this section, we examine the effectiveness of P4-TINS on the bandwidth guarantee and management for a sliced TCP flow that coexists with a non-sliced TCP flow (Section III-C) or a non-sliced UDP flow (Section III-D). As the bandwidth setting in the case study of Section III, two slice configurations $S(7, 8)$ and $S(2, 10)$ are set for the sliced TCP flow during the intervals I(20, 39) and I(40, 59), respectively. For the other times, no slice is enabled.

1) One Sliced TCP Flow, One Non-Sliced TCP Flow: With P4-TINS, the sliced TCP flow can obtain 7.47 Gbps and 5.71 Gbps throughput on average with the slice configurations $S(7, 8)$ and $S(2, 10)$, respectively, as shown in the upper part of Figure 14(a). In contrast with the default setting where the sliced TCP flow gets only 7.17 Gbps and 2.14 Gbps, P4-TINS can enable the sliced TCP flow to share more residual bandwidth 0.30 Gbps and 3.57 Gbps by 1.76x and 25.5x, respectively, while guaranteeing its CIR. It shows that separating the queue of sliced TCP flows from that of non-sliced TCP flows indeed takes effective. On the other hand, the non-sliced TCP flows with P4-TINS have average throughput 2.53 Gbps and 4.29 Gbps for those two slice configurations, respectively, as shown in the lower part of Figure 14(a).

We observe that the sliced TCP flow gets beyond-guaranteed bandwidth 0.47 Gbps and 3.71 Gbps with those two slice configurations, respectively. The amounts of the beyond-guaranteed bandwidth are almost proportional to those of the non-guaranteed, allowable bandwidth (i.e., PIR - CIR) from the slice configurations. The former's ratio is $3.71/0.47 = 7.89$, whereas the latter's ratio is $(10 - 2)/(8 - 7) = 8$. It is reasonable, since the non-guaranteed, allowable bandwidth represents how much traffic can be passed to the low-priority queues in P4-TINS and the beyond-guaranteed bandwidth comes from the low-priority queues; however, the small gap can be attributed to the dynamics of the TCP congestion control operations.

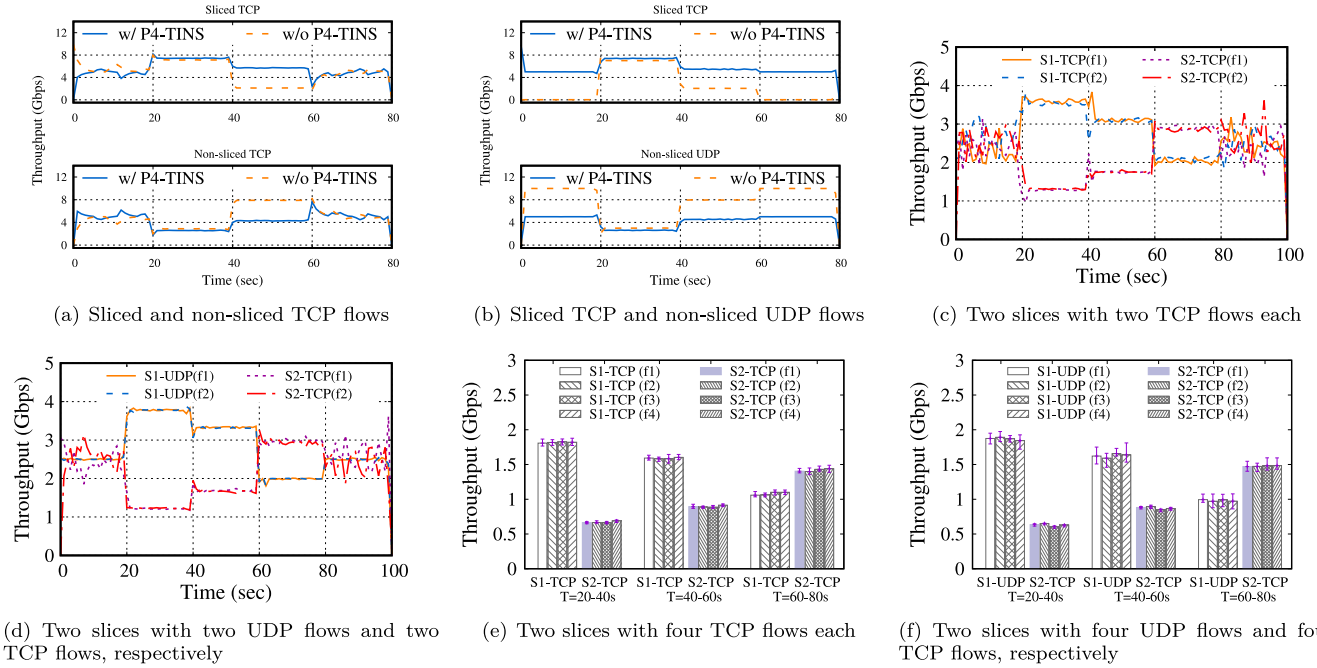
Note that although the weights are equal among low-priority queues in the experiment, the traffic amounts going to the low-priority queues from those sliced and non-sliced flows are dynamic due to the TCP congestion control operations. So, this experiment does not have the similar result observed for UDP traffic in Section III-B.

2) One Sliced TCP Flow, One Non-Sliced UDP Flow: As shown in Figure 14(b), we observe that P4-TINS can protect the sliced TCP flow from the interference of the non-sliced aggressive UDP flow by not only guaranteeing its CIR but also grabbing beyond-guaranteed bandwidth 0.39 Gbps and 3.46 Gbps from those two slice configurations, respectively. Without P4-TINS, the sliced TCP flow obtains only 7.17 Gbps and 2.14 Gbps. P4-TINS allows the sliced TCP flow to receive more residual bandwidth 0.22 Gbps and 3.32 Gbps by 1.29x and 23.71x, respectively. Compared with the above case, it receives a bit less residual bandwidth, because the non-sliced UDP flow can keep a static traffic amount going to the low-priority queue but the amount from the non-sliced TCP flow is dynamic due to its congestion control.

D. Multi-Slice Case With Multiple Flows

We further examine the effectiveness of P4-TINS in various multi-slice, multi-flow cases in this section. For all the cases, we consider two kinds of traffic flows combination: all slices with TCP flows, and two halves of slices with UDP and TCP flows, respectively. For clarity, we use a notation Sx-y(z) to represent Flow z in Slice x with protocol y (TCP or UDP); for example, S1-TCP and S2-UDP(f1) represent all the flows in Slice 1 with TCP and Flow 1 in Slice 2 with UDP, respectively.

To show that P4-TINS can dynamically guarantee each slice's CIR and allocate residual bandwidth fairly among slices, we vary CIR and PIR over time for each slice. For two-slice cases, S1 sets configurations S1(7, 8), S1(6, 8), and S1(2, 10) at the 20th, 40th, and 60th seconds, respectively, whereas S2 has S2(2, 10), S2(3, 10), and S2(5, 7), respectively. The slices are disabled before the 20th second and after the 79th second. For four-slice cases, S1, S2, S3, and S4 have configurations S1(1, 1), S2(1, 2), S3(1, 3), and S4(1, 4), respectively, during the interval I[20, 39], whereas they have S1(1, 4), S2(1, 3), S3(1, 2), and S4(1, 1), respectively, during the interval I[40, 59].



(a) Sliced and non-sliced TCP flows

(b) Sliced TCP and non-sliced UDP flows

(c) Two slices with two TCP flows each

(d) Two slices with two UDP flows and two TCP flows, respectively

(e) Two slices with four TCP flows each

(f) Two slices with four UDP flows and four TCP flows, respectively

Fig. 14. The effectiveness of P4-TINS on the bandwidth guarantee and management is examined in various cases. Slices 1 and 2 are abbreviated as S1 and S2, respectively.

Note that similar results are observed from other configurations, so we here use only the above settings to show the results due to limited space.

1) Two Slices With Two Flows Each: We conduct experiments in two two-slice cases. The first case (Case I) has two slices with two TCP flows each, whereas the second case (Case II) has one slice with two UDP flows and the other slice with two TCP flows; all the flows are with 10 Gbps. Their throughput results over time are shown in Figures 14(c) and 14(d), respectively. It is observed that the throughput of the flows can be dynamically constrained and each slice's CIR can be guaranteed according to slice configurations; within a slice, the throughput is fairly shared among two flows. We also observe that when a slice with two TCP flows has aggregate throughput higher than 4 Gbps (e.g., Slice S1-TCP in the interval I[20, 79] of Case I, Slice S2-TCP in the interval I[60, 79] of Case I, and Slice S2-TCP in the interval I[60, 79] of Case II), each of the TCP flows has fluctuating throughput, compared with each of the UDP flows in the UDP slice (i.e., S1-UDP). It can be attributed to the impact that some TCP packets being put into low-priority queues or dropped may cause timeout or fast retransmission from the congestion control.

Table I compares slice configuration and throughput results for these two cases. We have other two observations. First, Slice S1-UDP with UDP flows can grab more beyond-guaranteed bandwidth than Slice S2-TCP with TCP flows, when its non-guaranteed, allowable bandwidth (i.e., PIR - CIR) is not smaller than the overall beyond-guaranteed bandwidth (i.e., $mBW - \sum_i^{N_s} CIR_i$, where *mBW*, N_s, and CIR_i indicate maximum bandwidth, number of slices, and CIR of Slice i, respectively); for example, these two bandwidth values for S1-UDP in the interval I[40, 59] are 2 Gbps and 1 Gbps, respectively. Specifically, in Case II, S1-UDP can thus grab more beyond-guaranteed throughput (i.e., 0.58 Gbps, 0.66 Gbps, and 1.97 Gbps in those three configuration intervals, respectively), compared with S2-TCP, which gets only 0.39 Gbps, 0.31 Gbps, and 0.89 Gbps, respectively. The reason is that the UDP flows can generate constant traffic to the low-priority queues, but the traffic volume from the TCP flows fluctuates due to the congestion control. Second, for the slices with all TCP flows in Case I, the beyond-guaranteed throughput has a positive correlation with non-guaranteed, allowable bandwidth, but they are not proportional to each other possibly due to the dynamics of TCP congestion control.

TABLE I
CONFIGURATION AND THROUGHPUT RESULTS FOR TWO SLICES WITH TWO 10 GBPS FLOWS EACH

S1: two TCP flows; S2: two TCP flows (Gbps)				
Slices	Interval (sec)	20th-39th	40th-59th	60th-79th
S1-TCP	(CIR, PIR)	(7, 8)	(6, 8)	(2, 10)
	Ave. Throughput	7.12	6.18	4.14
S2-TCP	(CIR, PIR)	(2, 10)	(3, 10)	(5, 7)
	Ave. Throughput	2.58	3.49	5.73
S1: two UDP flows; S2: two TCP flows (Gbps)				
Slices	Interval (sec)	20th-39th	40th-59th	60th-79th
S1-UDP	(CIR, PIR)	(7, 8)	(6, 8)	(2, 10)
	Ave. Throughput	7.58	6.66	3.97
S2-TCP	(CIR, PIR)	(2, 10)	(3, 10)	(5, 7)
	Ave. Throughput	2.39	3.31	5.89

2) Two Slices With Four Flows Each: We further consider two two-slice cases with four 2 Gbps flows each using the same slice configuration over time. The goal is to examine whether more flows in slices can make the slices share residual bandwidth more fairly, since more TCP flows may compensate each other and then alleviate the fluctuation of aggregate traffic volume from TCP flows. The measured throughput over time in those two cases are shown in Figures 14(e) and 14(f),

TABLE II
CONFIGURATION AND THROUGHPUT RESULTS FOR TWO SLICES WITH FOUR 2 GBPS FLOWS EACH

S1: four TCP flows; S2: four TCP flows (Gbps)				
Slices	Interval (sec)	20th-39th	40th-59th	60th-79th
S1-TCP	(CIR, PIR)	(7, 8)	(6, 8)	(2, 10)
	Ave. Throughput	7.27	6.36	4.32
S2-TCP	(CIR, PIR)	(2, 10)	(3, 10)	(5, 7)
	Ave. Throughput	2.67	3.58	5.67
S1: four UDP flows; S2: four TCP flows (Gbps)				
Slices	Interval (sec)	20th-39th	40th-59th	60th-79th
S1-UDP	(CIR, PIR)	(7, 8)	(6, 8)	(2, 10)
	Ave. Throughput	7.48	6.51	3.94
S2-TCP	(CIR, PIR)	(2, 10)	(3, 10)	(5, 7)
	Ave. Throughput	2.52	3.48	5.90

respectively; the average throughput result is summarized in Table II. As consistent with the result of the last experiment, the CIRs of all the slices can be guaranteed, and the flows in each slice can fairly share the slice's throughput with only small difference ratios 0.98%-4.07% and 0.92%-7.37% for those two cases, respectively.

For the sharing of residual bandwidth, we have two main observations. First, the beyond-guaranteed throughput values of the slices in Case I still have positive correlations with non-guaranteed, allowable bandwidth; most of the slices can achieve higher throughput than those in the two-slice case with only two flows, because four TCP flows within a slice can compensate each other to generate more packets to the low-priority queues.

Second, S1-UDP and S2-TCP in Case II can fairly share the residual bandwidth with only up to 4% difference. Specifically, those three intervals (i.e., I[20, 39], I[40, 59], and I[60, 79]) with different configurations have residual bandwidth amounts, 1 Gbps, 1 Gbps, and 2 Gbps, respectively. The reason why the last interval has only 2 Gbps residual bandwidth but not 3 Gbps is that S2-TCP has 7 Gbps PIR which gives 3 Gbps guaranteed bandwidth to S1-UDP. As shown in the lower part of Table II, S1-UDP obtains residual bandwidth, 0.48 Gbps, 0.51 Gbps, and 0.94 Gbps, respectively; S1-TCP receives 0.52 Gbps, 0.48 Gbps, 0.90 Gbps, respectively. Compared with the case in Section V-D1, these two slices can have better fairness in sharing residual bandwidth, since the four TCP flows in S2-TCP are more aggressive than those two TCP flows only, to compete with S1-UDP; S1-UDP and S2-TCP are matched in the traffic intension towards the low-priority queues.

3) Four Slices With Two Flows Each: We next examine the effectiveness of P4-TINS on two four-slice cases. One has four TCP slices, whereas the other is with two UDP slices and two TCP slices; each slice has two 2 Gbps flows. Each slice is given 1 Gbps guaranteed bandwidth, but the slices have different limited bandwidth amounts. The total of the limited bandwidth amounts from all the four slices is set to the maximum bandwidth, 10 Gbps. Specifically, the four slices have configurations S1(1, 1), S2(1, 2), S3(1, 3), and S4(1, 4) in the interval I[20, 39], whereas they have S1(1, 4), S2(1, 3), S3(1, 2), and S4(1, 1) in the interval I[40, 59]. The configuration and throughput results of those two cases are summarized in Table III.

TABLE III
CONFIGURATION AND THROUGHPUT RESULTS FOR FOUR SLICES WITH TWO 2 GBPS FLOWS EACH

S1/S2/S3/S4: Two TCP flows each (Gbps)			
Slices	Interval (sec)	20th-39th	40th-59th
S1-TCP	(CIR, PIR)	(1, 1)	(1, 4)
	Ave. Throughput	0.99	3.97
S2-TCP	(CIR, PIR)	(1, 2)	(1, 3)
	Ave. Throughput	2.00	3.03
S3-TCP	(CIR, PIR)	(1, 3)	(1, 2)
	Ave. Throughput	3.00	1.99
S4-TCP	(CIR, PIR)	(1, 4)	(1, 1)
	Ave. Throughput	3.99	0.99
S1/S3: Two UDP flows each; S2/S4: Two TCP flows each (Gbps)			
Slices	Interval (sec)	20th-39th	40th-59th
S1-UDP	(CIR, PIR)	(1, 1)	(1, 4)
	Ave. Throughput	0.99	3.62
S2-TCP	(CIR, PIR)	(1, 2)	(1, 3)
	Ave. Throughput	2.01	3.02
S3-UDP	(CIR, PIR)	(1, 3)	(1, 2)
	Ave. Throughput	2.99	1.86
S4-TCP	(CIR, PIR)	(1, 4)	(1, 1)
	Ave. Throughput	3.99	0.99

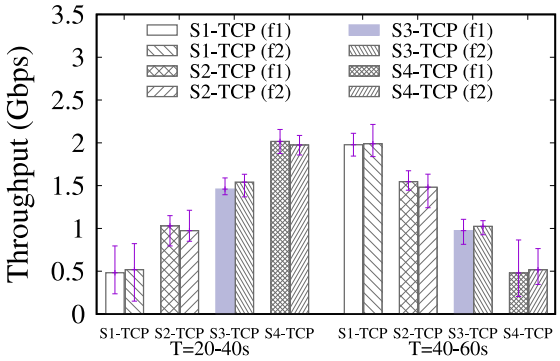


Fig. 15. P4-TINS for four slices with two flows each: 2 UDP slices and 2 TCP slices.

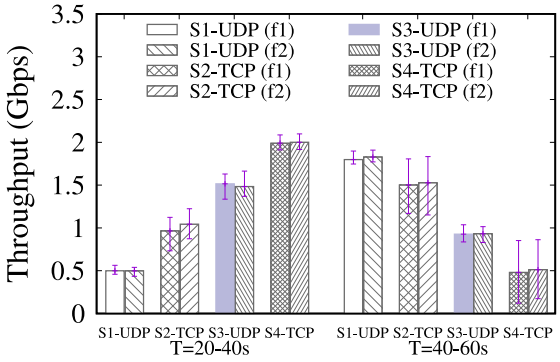


Fig. 16. P4-TINS for four slices with two flows each: 4 TCP slices.

We make three observations. First, all the TCP slices can receive residual bandwidth as much as possible, i.e., the average throughput is roughly equal to the limited bandwidth with a maximum difference 0.03 Gbps. For example, when S2-TCP switches from the setting (1, 2) in the interval I[20, 39] to (1, 3) in I[40, 59], its average throughput changes from 2.00 Gbps to 3.03 Gbps and from 2.01 Gbps to 3.02 Gbps in those two cases, respectively. Second, the flows in each slice can fairly share the slice's bandwidth with only small differences ranging from 0.54% to 8.23%, as shown in Figures 16(a) and 16(b).

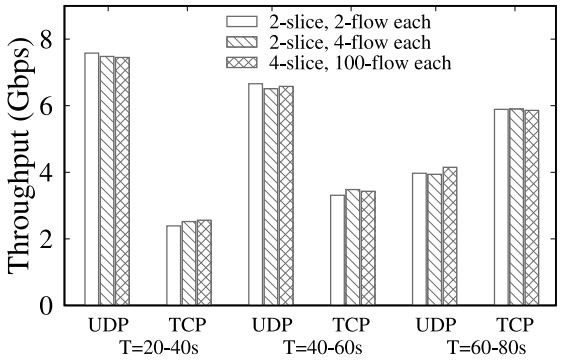


Fig. 17. Average throughput of UDP and TCP slices with 2, 4 and 100 flows each.

Third, the UDP slices in the interval I[20, 39] in the second case can achieve the average throughput as much as their limited bandwidth, but they cannot achieve it in the interval I[40, 59]. Different from the two-flow case with 10 Gbps flows in Section V-D1, each flow is given only 2 Gbps in this case. Specifically, S1-UDP with PIR 4 Gbps and S3-UDP with PIR 2 Gbps can achieve only the average throughput values, 3.62 Gbps and 1.86 Gbps, respectively. The root cause is that the low-priority queue assigned to UDP flows cannot accommodate transient traffic amount from the sliced UDP flows, which generate non-slice traffic 3 Gbps and 1 Gbps from S1-UDP and S3-UDP, respectively. It can cause UDP packets to be dropped, whereas although some TCP packets may be also dropped, TCP flows have the retransmission mechanism. Note that the maximum size of each low-priority queue is 25,000 packets. With MTU 1500 bytes, it can support 300 Mb. When the traffic is generated every 100 ms, the queue can support only 3 Gbps, which is smaller than the total non-slice traffic amounts, 4 Gbps, in the second interval.

E. Multi-Slice Case With Many Flows

We finally evaluate P4-TINS in three multi-slice cases with many flows: 4 slices with 100 flows each, 4 slices with 500 flows each and 20 slices with 100 flows each. We focus on the coexistence of UDP and TCP slices in these cases.

1) 4 Slices With 100 Flows Each: We examine whether P4-TINS can still work for many flows by considering the slice configuration similar to the two-slice cases with few flows in Sections V-D1 and V-D2 for a four-slice case with 100 flows per slice. In those two-slice cases, Slice S1-UDP has configurations S1(7, 8), S1(6, 8), and S1(2, 10) in the three intervals of I[20, 39], I[40, 59], and I[60, 79], respectively, whereas Slice S2-TCP has S2(2, 10), S2(3, 10), and S2(5, 7), respectively. This four-slice case consists of two UDP slices and two TCP slices. The two UDP slices equally share the CIR/PIR of the S1-UDP configurations, so each has configurations S(3.5, 4), S(3, 4), and S(1, 5), respectively; each of the two TCP slices also has an equal CIR/PIR share of the S2-TCP configurations.

We compare the result of this 100-flow case with those aforementioned 2-flow and 4-flow cases in Figure 17. It shows the average throughput of aggregate UDP slices and that of TCP slices for those three intervals with different slice configurations. It is observed that each of the 4 slices can successfully obtain guaranteed CIR for all the three sets of slice configurations. For the residual bandwidth, the result of the 100-flow case is similar to that of 2-flow and 4-flow cases. Specifically, compared with the latter cases, the UDP and TCP slices of the 100-flow case have only small differences, no more than 5.06% and 6.64%, respectively, in average throughput.

2) 4 Slices With 500 Flows Each: We next increase the number of flows to 500 in each slice. By considering two sets of slice configurations, two UDP slices have configurations as S1(1, 2) and S3(1, 4) in the interval I[20, 39], and are then assigned S1(1, 2) and S3(3, 4) in I[40, 59]; two TCP slices S2 and S4 have the same slice configurations as S1 and S3. Table IV summarizes the slice configuration and average throughput of each slice in each test interval.

The result shows that P4-TINS can work effectively for a great number of flows. It is seen that all the slices can successfully obtain guaranteed CIR. No obvious interference between them is observed; the beyond-guaranteed throughput of a slice is still positively correlated with the slice's non-guaranteed, allowable bandwidth. Specifically, S3-UDP and S4-TCP slices obtain the beyond-guaranteed throughput 2.24 Gbps and 2.04 Gbps, respectively, in the interval I[20, 39]; when their non-guaranteed, allowable bandwidth is reduced from 3 Gbps to 1 Gbps in the interval I[40, 59], their beyond-guaranteed throughput values decrease to 0.55 Gbps and 0.62 Gbps, respectively. Although UDP and TCP slices have the same slice configurations, their bandwidth shares are not completely equal due to the dynamics of many flows and TCP congestion control.

3) 20 Slices With 100 Flows Each: We next investigate the effectiveness of P4-TINS in a case of many slices with many flows. This case consists of 10 TCP slices and 10 UDP slices with 100 flows per slice. The slice configuration for each of the TCP/UDP slices is S(0.2, 1) (i.e., the aggregate slice configuration of each slice type is S(2, 10)) in the first test interval I[20, 39]; then, in the second test interval I[40, 59], that for each TCP slice remains the same, but that for each UDP slice becomes S(0.4, 1). It is observed that each slice can obtain bandwidth no less than its CIR. The bottom part of Table IV summarizes the aggregate slice configuration and average throughput of each slice type in each test interval. The result shows that the aggregate TCP slices almost achieve the same beyond-guaranteed throughput as the aggregate UDP slices in both of the test intervals; no interference happens between UDP and TCP slices. It confirms the effectiveness of P4-TINS with many slices and flows.

VI. Discussion

In this section, we discuss whether P4-TINS can be applied to other P4 chips and other transport protocols.

Other P4 chips: The P4 operation considered in this work includes a standard P4 feature, meter, and a conventional default metering algorithm, trTCM, so it is expected that these two components are also used on other P4 chips and similar results can be observed (i.e., similar issues in the case study

TABLE IV
CONFIGURATION AND THROUGHPUT RESULTS FOR A 4-SLICE CASE WITH 500 FLOWS PER SLICE AND A 20-SLICE CASE WITH 100 FLOWS PER SLICE

S1/S3: 500 UDP flows each; S2/S4: 500 TCP flows each (Gbps)			
Slices	Interval (sec)	20th-39th	40th-59th
S1-UDP	(CIR, PIR)	(1, 2)	(1, 2)
	Ave. Throughput	1.75	1.44
S2-TCP	(CIR, PIR)	(1, 2)	(1, 2)
	Ave. Throughput	1.96	1.39
S3-UDP	(CIR, PIR)	(1, 4)	(3, 4)
	Ave. Throughput	3.24	3.55
S4-TCP	(CIR, PIR)	(1, 4)	(3, 4)
	Ave. Throughput	3.04	3.62
10 UDP and 10 TCP slices with 100 flows each (Gbps)			
Slices	Interval (sec)	20th-39th	40th-59th
UDP	(CIR, PIR)	(2, 10)	(4, 10)
	Ave. Throughput	5.00	6.01
TCP	(CIR, PIR)	(2, 10)	(2, 10)
	Ave. Throughput	5.00	3.99

and the effectiveness of P4-TINS). However, P4-TINS cannot be applied to some P4 chips without supporting the priority queues, which can only limit bandwidth but cannot guarantee bandwidth or manage residual bandwidth.

Other transport protocols: The major issue caused by the interference presented in this paper lies in a mismatched operation between high-priority and low-priority queues, and can thus hurt the congestion control of sliced TCP flows. P4-TINS can resolve this issue using separate low-priority queues for different flow types and can be extended to the other transport protocols with congestion control. Take two important transport protocols, QUIC and GTP, as examples. For the QUIC flows with a new TCP-like congestion control mechanism over UDP, the sliced and non-sliced QUIC flows can use the low-priority queues of sliced and non-sliced TCP, respectively, to prevent the interference from non-sliced flows and native UDP flows. For the GTP flows, which may contain both UDP and TCP packets in the inner IP packets, the GTP packets should be dispatched to different low-priority queues based on their inner packet types. However, these two protocols require a new identification module of traffic types, which is different from the conventional one purely based on transport headers, since identifying QUIC packets needs to check UDP payload, and differentiating GTP/UDP and GTP/TCP packets needs to examine their inner transport headers. This new identification module may cause more overhead while the interference can be prevented on the QUIC and GTP flows. We will consider these two important transport protocols in the future work.

VII. CONCLUSION AND FUTURE WORK

Applying bandwidth guarantee and management flexibly to network slices requires the support of programmable switches. Current popular programmable switches are P4-based, so for easy deployment, we seek to propose a P4-compatible solution. To this end, we conduct a case study to examine the effectiveness of the bandwidth guarantee and management on a COST P4 switch with the Intel/Barefoot Tofino P4 chip. We discover that a traffic flow in a slice can be interfered by another in the same slice or outside the slice. It can cause the traffic flow to receive unfair residual bandwidth. Moreover, inappropriate bucket sizes can prevent sliced TCP flows from getting guaranteed bandwidth. We propose P4-TINS to resolve the interference by isolating different types of traffic flows among multiple priority queues while setting an appropriate bucket size. Our prototype on the COTS P4 switch confirms the effectiveness of the bandwidth guarantee in all the cases and achieves fair residual bandwidth share among coexistent UDP/TCP network slices without interference.

In the future work, we have two major research directions. First, we seek to provide more fine-grained fairness among low-priority queues by assigning them weights based on the number of their active flows or their different traffic demands, instead of giving them equal weights all the time in this work. Second, we will apply P4-TINS to achieving network slicing in the user plane of 5G mobile networks and examine its performance.

REFERENCES

[1] *5G;System Architecture for the 5G System (3GPP TS 23.501 version 15.2.0 Release 15)*, ETSI Standard TS 123 501, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf

[2] Q. Wang *et al.*, "Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 444–453, Jun. 2019.

[3] N. K. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating fair queueing on reconfigurable switches," in *Proc. USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2018, pp. 1–16.

[4] N. K. Sharma *et al.*, "Programmable calendar queues for high-speed packet scheduling," in *Proc. USENIX Symp. Netw. Syst. Des. Implement. (NSDI')*, 2020, pp. 685–700.

[5] A. Sivaraman *et al.*, "Programmable packet scheduling at line rate," in *Proc. Conf. ACM Spec. Interest Group Data Commun. (SIGCOMM)*, 2016, pp. 44–57.

[6] V. Shrivastav, "Fast, scalable, and programmable packet scheduler in hardware," in *Proc. ACM Spec. Interest Group Data Commun. (SIGCOMM)*, 2019, pp. 367–379.

[7] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "SP-PIFO: Approximating push-in first-out behaviors using strict-priority queues," in *Proc. USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2020, pp. 59–76.

[8] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proc. ACM SIGCOMM Symp. SDN Res. (SOSR)*, 2017, pp. 164–176.

[9] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. ACM SIGCOMM Symp. SDN Res. (SOSR*, 2016, pp. 1–12.

[10] B. Turkovic, F. Kuipers, N. van Adrichem, and K. Langendoen, "Fast network congestion detection and avoidance using P4," in *Proc. Workshop Netw. Emerg. Appl. Technol.*, 2018, pp. 45–51.

[11] Q. Duan, "Network-as-a-service in software-defined networks for end-to-end QoS provisioning," in *Proc. Int. Conf. Wireless Opt. Commun. Conf. (WOCC)*, 2014, pp. 1–5.

[12] C. Morin, G. Texier, and C.-T. Phan, "On demand QoS with a SDN traffic engineering management (STEM) module," in *Proc. Int. Conf. Netw. Serv. Manag. (CNSM)*, 2017, pp. 1–6.

[13] A. Mirchev, "Survey of concepts for QoS improvements via SDN," in *Proc. Future Internet (FI) Innov. Internet Technol. Mobile Commun. (IITM)*, vol. 33, 2015, pp. 33–39.

[14] J. Moeyersons *et al.*, "Enabling emergency flow prioritization in SDN networks," in *Proc. Int. Conf. Netw. Serv. Manag. (CNSM)*, 2019, pp. 1–8.

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

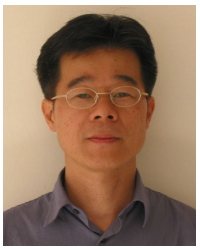
[16] P. M. Mohan, D. M. Divakaran, and M. Gurusamy, "Performance study of TCP flows with QoS-supported OpenFlow in data center networks," in *Proc. IEEE Int. Conf. Netw. (ICON)*, 2013, pp. 1–6.

[17] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers, "Providing bandwidth guarantees with OpenFlow," in *Proc. IEEE Symp. Commun. Veh. Technol. (SCVT)*, 2016, pp. 1–6.

[18] N. Kitsuwan and E. Oki, "Traffic splitting technique using meter table in software-defined network," in *Proc. IEEE Int. Conf. High Perform. Switch. Routing (HPSR)*, 2016, pp. 108–109.
[19] Z. Yu *et al.*, "Programmable packet scheduling with a single queue," in *Proc. Conf. ACM Spec. Interest Group Data Commun. (SIGCOMM)*, 2021, pp. 179–193.
[20] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.
[21] "$P4_{16}$ Portable Switch Architecture (PSA)." The P4.org Architecture Working Group. 2018. [Online]. Available: https://p4.org/p4-spec/docs/PSA-v1.1.0.html
[22] J. Heinanen and R. Guerin, "A two rate three color marker," IETF, RFC 2698, 1999.
[23] Y.-W. Chen, L.-H. Yen, W.-C. Wang, C.-A. Chuang, Y.-S. Liu, and C.-C. Tseng, "P4-enabled bandwidth management," in *Proc. Asia–Pacific Netw. Oper. Manag. Symp. (APNOMS)*, 2019, pp. 1–5.
[24] K. Tokmakov, M. Sarker, J. Domaschka, and S. Wesner, "A case for data centre traffic management on software programmable Ethernet switches," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, 2019, pp. 1–6.
[25] H. Harkous, C. Papagianni, K. De Schepper, M. Jarschel, M. Dimolianis, and R. Pries, "Virtual queues for P4: A poor man's programmable traffic manager," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2860–2872, Sep. 2021.
[26] S.-Y. Wang, H.-W. Hu, and Y.-B. Lin, "Design and implementation of TCP-friendly meters in P4 switches," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1885–1898, Aug. 2020.
[27] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2014, pp. 1–6.
[28] "$P4_{16}$ Language Specification Version 1.1.0." The P4 Language Consortium. 2018. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.1.0-spec.html
[29] "Flowgrind—TCP Traffic Generator." [Online]. Available: https://github.com/flowgrind/flowgrind (Accessed: Mar. 2022).
[30] "P4Runtime Specification Version 1.0.0." The P4.org API Working Group. 2019. [Online]. Available: https://p4.org/p4runtime/spec/v1.0.0/P4Runtime-Spec.html

Yan-Wei Chen received the M.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2020. His research interests include software defined networking and programmable P4 switch.

Chi-Yu Li (Member, IEEE) received the B.S. and M.S. degrees from the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, and the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2015. He is currently an Associate Professor with the Department of Computer Science, National Yang Ming Chiao Tung University. His research interests include wireless networking, mobile networks and systems, and network security. He received the Award of MTK Young Chair Professor in 2016, the two MOST grants for Excellent Young Scholar in 2017 and 2021, the three MOST Futuretech Awards in 2020, 2021, and 2021, and the Best Paper Award in IEEE CNS 2018.



Chien-Chao Tseng received the M.S. and Ph.D. degrees in computer science from the Southern Methodist University, Dallas, TX, USA, in 1986 and 1989, respectively. He joined the Department of Computer Science with National Yang Ming Chiao Tung University, Hsinchu, Taiwan, in 1989 and is currently a Professor of the Department. His research interests include software defined networks, network function virtualization, wireless Internet, and heterogeneous networks. He is the P.I. of the SDNFV Orchestration for 5G project at the Center for Open Intelligent Connectivity sponsored by the Higher Education Sprout Project by the Ministry of Education in Taiwan.

Min-Zhi Hu received the B.S. degree in computer science from National Chung Cheng University, Chiayi, Taiwan, in 2020. He is currently a master student with the Department of Computer Science, National Yang Ming Chiao Tung University. His research interests include software defined networking and programmable P4 switch.