P4-Enabled Bandwidth Management

Yan-Wei Chen, Li-Hsing Yen, Wei-Cheng Wang, Cheng-An Chuang, Yu-Shen Liu, and Chien-Chao Tseng
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

***Abstract*—As the next generation network is supposed to support diverse service requirements, managing Quality of Service (QoS) is a crucial part of it. QoS guarantees have long been deemed too complicated until the emergence of software-defined networking (SDN) and widely adopted standard OpenFlow. Recently, Programming Protocol-independent Packet Processors (P4) has gained much attention because of its features like programmable data plane and independent protocol and platform. It is anticipated that the high flexibility of P4 can enhance the QoS control for production networks.**

In this paper, we show a design of bandwidth management for QoS with SDN and P4-programmable switch. The design classifies packets into different categories based on their QoS demands and usages, which are then disaggregated by a two-level priority queue. Experiments with P4 switch shows that the proposed design not only effectively limits the maximum allowed rate but also guarantees the minimum bandwidth of each traffic flow. As such, the design can maximize bandwidth utilization and serves a building block for network slicing.

***Index Terms*—Quality of Services, Bandwidth Management, Software Defined Networking, Programming Protocol-independent Packet Processors**

I. Introduction

A fundamental challenge in the next generation network is to fully utilize computation and networking resources to provide services that meet diverse Quality of Services (QoS) requirements [1]. Such requirements are not only essential to applications running on end devices but also crucial to the infrastructure for provisioning network virtualization. As a promising building block for network virtualization, software-defined networking (SDN) offers an effective approach to providing end-to-end QoS management [2] with a global view of the network topology on the centralized control plane.

There have been many researches on QoS provisioning or bandwidth management on SDN [3]–[5], mostly with the widely adopted SDN southbound protocol OpenFlow [7]. However, OpenFlow-compliant switches built with an application-specific integrated circuit (ASIC) technology provide specific and rather restrictive functionalities. More specifically, the number of pipeline stages, the number of supported header formats, and the types and numbers of actions that could apply are all limited. Therefore, the implementation would be clumsy when we want to apply multiple policies to a flow as a means to achieve differentiated QoS.

As a way to provide flexible functionality, Programming Protocol-independent Packet Processors (P4) [8] has been introduced. P4 technology allows customized switch pipelines so that the switch is not tied to specific header formats and actions.

In this paper, we present a P4-enabled bandwidth manager for end-to-end QoS provisioning. The main features of the manager include peak rate limitation and minimal bandwidth guarantee. The design also includes a set of REST APIs for third-party applications to perform meticulously bandwidth management. The design has been implemented with Open Network Operating System (ONOS) [9], which provides more comprehensive support [10] for P4 than other SDN controllers. The work here serves as an essential building block for network slicing with QoS awareness [1].

The rest of this paper is organized as follows. We introduce related work and mechanisms of P4 traffic management in the next section. Section III will present our design of bandwidth management. Section IV will provide some traffic evaluation. Finally, Section V gives a brief conclusion.

II. Background and Related Work

A. QoS support in OpenFlow

In SDN, a typical means of QoS provisioning is to monitor the bandwidth usage for each flow and perform appropriate action on the flow based on the bandwidth usage. A set of actions, each targeting at a specific range of bandwidth usage, comprise a policy that could apply to a designated flow. OpenFlow implements a policy with Meter and Queue. Queue enables the configuration of a multi-level priority queue associated with an egress port. With this support, packets toward to the same egress port can have different priorities when being forwarded.

OpenFlow 1.3 [11] introduced the Meter table, which measures and controls ingress traffic on a per-flow basis. Meter entry, as shown in Fig. 1, is attached to a flow entry and consists of one or more Meter bands. Each Meter band corresponds to a specific policy that will apply to the flow when the current bandwidth usage of the flow is within some designated range. More specifically, a Meter band defines a specific rate value and an associated operation called band type. When the measured rate of a flow is r, the Meter band that has the highest rate among all Meter bands with a rate lower than r will be selected. For example, suppose that there are three bands configuring rates 30 Mbps, 60 Mbps, and 90 Mbps, respectively, for a flow. If the current rate of the flow is 70 Mbps, the operation associated with the band of 60 Mbps will be executed. Currently, OpenFlow supports only two band types (operations): dropping packets to enforce rate limitation [3] and setting DSCP fields for packets classification [4], [5].

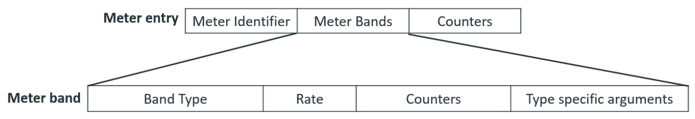

Fig. 1. OpenFlow Meter Entry

B. QoS support in P4

Compared with OpenFlow, Meter in P4 is more flexible as it does not tie to fixed band type. With Meter, P4 first classifies an ingress packet and then executes an action on the packet based on the classification result [6]. The P4 specification suggests using Two Rate Three Color Marker (trTCM) [12] as a default mechanism for packet classification. The trTCM gauges a packet stream and classifies packets with two user-configurable parameters, namely Peak Information Rate (PIR) and Committed Information Rate (CIR). PIR is the highest allowable data rate for a flow, while CIR is the lowest rate for the flow guaranteed by the network. The trTCM mechanism marks packets based on the gauged rate and the setting of PIR and CIR on a per-flow basis. A packet is marked red if forwarding it will cause a flow data rate exceeding the PIR. If the resulting flow data rate will not exceed PIR, the packet is marked yellow if the resulting rate will exceed CIR and green otherwise. After the classification, P4 makes use of per-packet metadata to keep the identified color.

P4 supports two types of Meters. One type called Direct Meter is used inside a specified flow table. When a flow entry of a flow table is matched, the associated action may trigger a Direct Meter in the same table to execute the packet classification with the identified color kept in the metadata field. The other type called Indirect Meter is an array of Meters to be referenced by one or more flow tables. Different Meter cells in Indirect Meter may be configured with different trTCM parameters. When a flow entry is matched, the associated action may trigger an execution of the packet classification with an index to some cell in the Indirect Meter. Both Direct Meter and Indirect Meter take user-specified trTCM parameters for the packet classification, as such a control plane entity can dynamically modify these parameters for multiple band requirements on a flow.

C. OpenFlow Meter vs P4 Meter

There are some differences between the OpenFlow Meter and P4 Meter:

- Number of Bands: An OpenFlow Meter may contain an arbitrary number of bands while a P4 Meter needs only two bands for the mechanism of trTCM implemented in the switch.
- Creation of Meter: OpenFlow controller can dynamically create a Meter instance and configure the Meter Bands. In contrast, Meters in P4 are statically defined by the P4 program; Controller can only dynamically change the trTCM parameters (PIR and CIR) of an existing Meter.
- Actions of Meter: OpenFlow Meter supports only two actions (dropping and marking DSCP in the packet header), but P4 Meter does not have that limitation.

There have been some QoS researches on per-flow traffic classification with the support of Meters in OpenFlow [4], [5]. These researches used DSCP bit in the IP header to classify packets. An intrinsic issue with the use of DSCP is that DSCP remark operation will modify the IP header. Consequently, when we have multiple DiffServ policers applying to the same flow, a modification on the IP header by one policer may interfere with the operations of others, causing policer interference problem. This problem could be avoided using P4 Meter because the classification result of a P4 Meter is kept in a dedicated metadata field. With this feature, we can define multiple actions, one for each policer, such that the classification result needed by each policer is kept separately and hidden from others.

Another limitation with OpenFlow Meter is lacking flexibility because of vendors fixed-functionality ASIC implementations. As such, managing a traffic flow by classification result may demand a loopback connection for header evaluation or a rewriting between different tables. With P4 switches, we may extend switch pipelines to add new management rules without extra loopback connection or a rewiring between different tables. This implementation is more straightforward and could save table entries.

III. P4-based Bandwidth Management

To resolve the policer interference problem and to circumvent the limitation by fixed-functionality ASIC implementation, we design a bandwidth manager on ONOS with a customized P4 pipeline for traffic engineering. As shown in Fig. 2, the main concept of our design is to classify flow traffic into three types, namely guaranteed traffic, best effort traffic, and abandon traffic. Guaranteed traffic receives high priority when being forwarded to an egress port, which serves as a means to allocate a portion of the bandwidth of the forwarding link to fulfill the guaranteed bandwidth. Best effort traffic receives low forwarding priority as the traffic can only utilize unallocated bandwidth of the forwarding link. Abandon traffic is simply dropped by the switch as forwarding it will give the flow a rate exceeding the limited bandwidth.

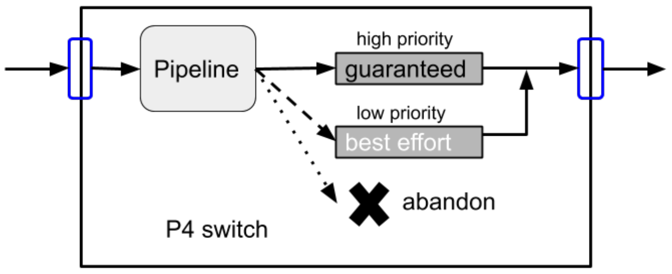

Fig. 2. Design Concept

However, if guaranteed traffic and best effort traffic are all fed into the same output queue, it is difficult to give priority to guaranteed traffic over best effort traffic. Therefore, we use a two-level priority queue scheduler for the egress port. Guarantee traffic goes to high-priority queue while best-effort traffic goes to low-priority queue. Packets in the low-priority queue will not be served unless the high-priority is empty.

The architecture of the implementation is shown in Fig. 3. It consists of a component in the control plane and another component in the data plane. In the control plane, we developed the Bandwidth Manager as an application running on ONOS to perform resource management and handle external bandwidth management requests. In the data plane, we designed a new P4 program Meter.p4 for P4 switch to perform per-flow traffic classification and traffic steering. Other existing-applications running on ONOS provide basic functionalities like forwarding and Proxy ARP. All ONOS applications use P4Runtime as a southbound API to communicate with P4 switches.

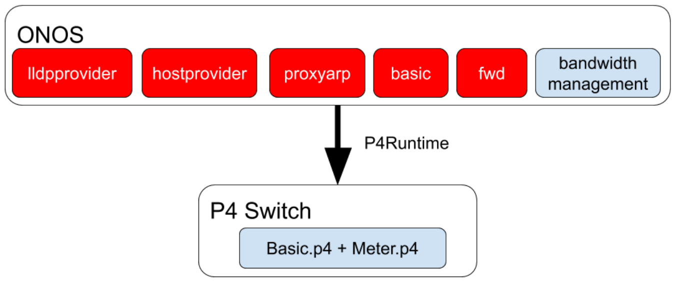

Fig. 3. The architecture of P4-Enabled Bandwidth Management with ONOS

A. *ONOS Bandwidth Manager*

As shown in Fig. 4, the kernel of Bandwidth Manager is Slice Manager, which manages slices, flows, and bandwidth resource. Each flow is assigned to a single slice, and all flows assigned to the same slice share whole bandwidth of the slice.

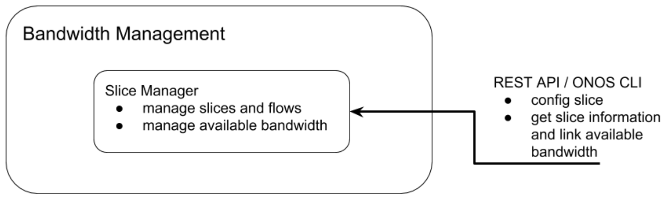

Fig. 4. Bandwidth Manager Application

Slice Manager provides a REST API and CLI commands for external requests to access slice information and dynamically manage slices (adding or deleting a slice and adjusting the rate of an existing slice). The REST API can also accept a JSON format file, as shown in Fig. 5, to dynamically control slices and adjust the bandwidth rate parameters in a Meter cell. Each slice has the following parameters to configure:

- minRate: the guaranteed bandwidth,
- maxRate: the limited bandwidth,
- flows: user-defined field, e.g., a five-tuple field (source IP address, destination IP address, etc.), used to identify a flow or flows within a slice.

```
{
  "slices": [
    {
      "minRate": 10,
      "maxRate": 20,
      "flows": [
        {
          "src_ip": "10.0.1.1",
          "dst_ip": "10.0.3.1"
        },
        {
          "src_ip": "10.0.2.1",
          "dst_ip": "10.0.3.1"
        }
      ]
    },
    {
      "minRate": 1000,
      "maxRate": 2000,
      "flows": [
        {
          "src_port": 80
        }
      ]
    }
  ]
}
```

Fig. 5. Slice Configuration JSON Example

When Slice Manager receives a request through REST API for the addition of a new flow into a bandwidth slice, it first checks to see whether the request can be granted with available bandwidth. If the result is positive, Slice Manager allocates requested bandwidth to the flow and assigns it to the slice. Slice Manager also assigns an identifier to the flow and installs a rule on P4 switch for the identification of packets belonging to this flow. In addition, it also installs a rule which takes the flow identifier to trigger a Meter cell of the slice to perform classification and record the identified color. Finally, the forwarding policer will process the colored packets and apply the actions based on the traffic type.

B. *P4 Pipeline for Classification*

The P4 program Meter.p4 classifies flow packets and then applies rate limit or bandwidth guarantee actions accordingly. For packet classification, it takes trTCM parameters PIR as the limit bandwidth and CIR as the guaranteed bandwidth, as shown in Fig. 6. With this setting, Meter.p4 classifies packets into three categories, each with a distinct color, and the results are kept as metadata of the packets. When packets of a flow are classified as green, the bandwidth currently allocated to this flow is lower than its CIR. Therefore, Meter.p4 attempts offering the guaranteed bandwidth by setting high forwarding priority to these packets. When packets are in yellow, the allocated bandwidth is between CIR and PIR (or these parameters are unspecified). In this case, Meter.p4 provides these packets with best-effort packet delivery by allocating only residual bandwidth to them via low forwarding priority. For packets in red, the bandwidth currently allocated to the flow already exceeds PIR. Therefore, Meter.p4 simply drops these packets.

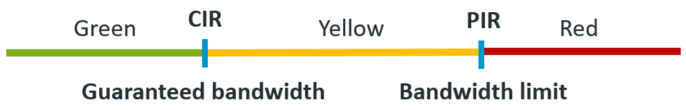

Fig. 6. Classifying packets into three different colors

The P4 pipeline design of our bandwidth manager is an extension to the ONOS built-in Basic pipeline [13]. The Basic pipeline provides fundamental data-plane functionalities of the switch, such as prefix matching, packet in, and output to the egress port. It is the Basic pipeline that provides switch features to our bandwidth manager. The Basic pipeline connects to our Meter.p4 pipeline, as shown in Fig. 7. Firstly, a slicing table is used to identify the slice of ingress packets and assign the identifier. This is done by matching the user-defined fields of packets within the corresponding slice, and we use the five-tuple field in this paper. All no-match packets will get the default slice identifier. The following is the classifier table, which takes the slice identifier as an index to a Meter cell in Indirect Meter. The target Meter cell uses the trTCM mechanism to perform a rate-based packet classification. Finally, the policer table will drop packets or set up forwarding priorities for the packets, depending on the classification result.

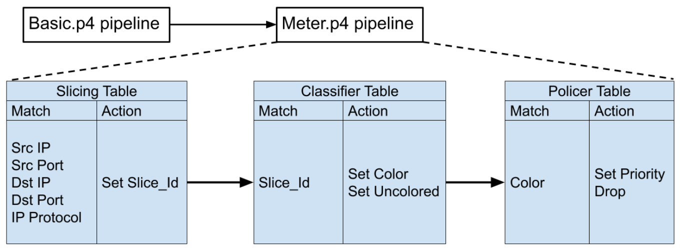

Fig. 7. Bandwidth Manager Pipeline

IV. Evaluation

We conducted some experiments using physical P4 switch to test our design. We measured the throughputs of flows under different settings of limited bandwidth and guaranteed bandwidth. We also compare the results between with and without the two-level priority queue.

A. Environment

Fig. 8 shows the testbed network for the experiments. There was an EdgeCore P4 switch (WEDGE100BF-32X) which uses Tofino chip of Barefoot. The Tofino chip is based on the Protocol-Independent Switch Architecture (PISA) and can be programmed by P4_16 [14]. The P4 switch is running Barefoot Capilano Software Development Environment (SDE) with customized pipeline and configured priority queue facility. An independent layer-2 switch is used to measure the data output rates of the P4 switch and server hosts. We used three server hosts running Ubuntu 16.04 based on Linux Kernel v4.15.0 with two 20-core Intel Xeon E5-2630 CPUs (3.1GHz). Each host is equipped with Intel X710 10GbE NICs with MTU set to 1500 bytes. Two of the hosts each was configured to generate 10-Gbps traffic from one port to create congestions on a 10-Gbps outgoing link of the P4 switch. The third host was used to receive traffic from two different ports. The bandwidth of each link in the testbed network is 10 Gbps.

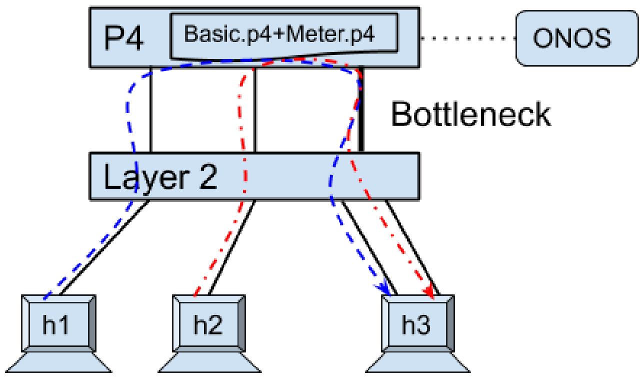

Fig. 8. Testbed

B. Rate Limitation

The first experiment was to test whether the Bandwidth Manager is effective in rate limitation. We generated a 10-Gbps UDP flow with the help of Iperf [15], and changed maxRate of this flow to 4, 8, 1, and 7.5 Gbps after 200, 400, 600, and 800 seconds, respectively. The gauged UDP throughput is shown in Fig. 9. It clearly shows that the throughputs were indeed limited by the settings of maxRate, which confirms the effectiveness of Bandwidth Manager in rate limitation.

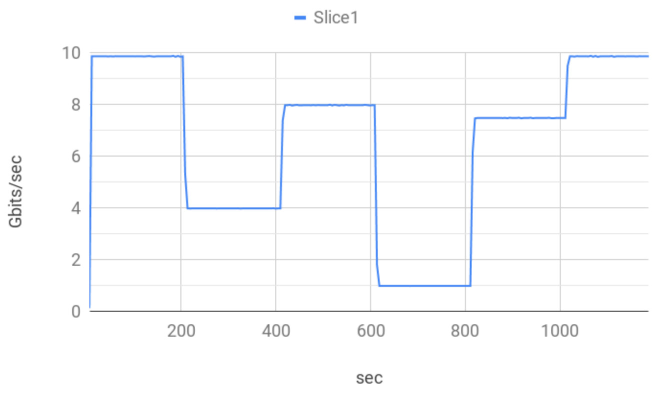

Fig. 9. Rate Limited Result

C. Minimal Bandwidth Guarantee

The second experiment was to test whether the Bandwidth Manager could guarantee minimum bandwidth. The experiment started with two 10-Gbps UDP flows, one from hosts h1 to h3 and the other from hosts h2 to h3. One flow belonging to Slice 1 had minRate set while the other serving as background traffic did not have any rate guarantee. We changed minRate to 3, 7, 0.8, and 6.5 Gbps after 200, 400, 600, and 800 seconds, respectively. Fig. 10 shows how the measured throughput of Slice 1 changed with time. We can see that Slice 1 received a throughout not lower than minRate at all time. The background traffic received residual bandwidth without any guarantee. The aggregated bandwidth of Slice1 and the background traffic was roughly 10 Gbps at all time.

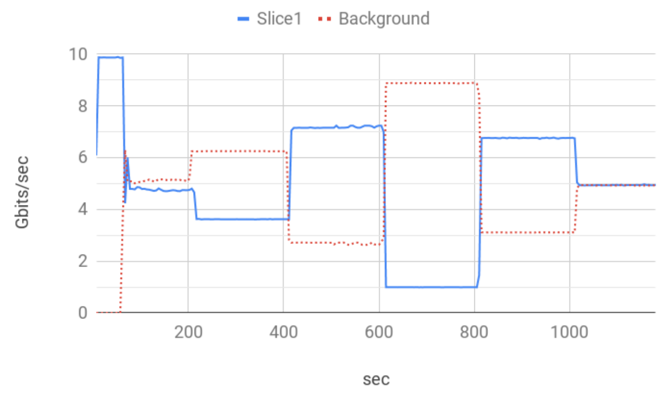

Fig. 10. Bandwidth Guaranteed Result

D. Impact of Two-Level Priority Queue

This section provides a comparison between queue-enabled and queue-disabled modes for bandwidth guarantee. We set up two different slices, namely Slice 1 and Slice 2. Slice 1 had a 10-Gbps UDP flow with minRate set to 6 Gbps and maxRate set to 10 Gbps. Slice 2 had a 4-Gbps UDP flow with both minRate and maxRate set to 4 Gbps. Slice 2 traffic started from the 200th second and ended at the 600th second.

When the two-level priority queue was not in use, the policier sent all traffic to a single output queue of the egress port. In this case, as can be seen from Fig. 11, the flow of Slice 2 failed to have the guaranteed bandwidth. With the use of the two-level priority queue, the throughput of Slice 2 was 4 Gbps as guaranteed.

V. Conclusion

In this paper, we present a bandwidth management design for P4 programmable switch. The design maximizes bandwidth utilization by allowing the best effort or non-guaranteed traffic to use unallocated bandwidth. On the other hand, when the aggregated input traffic exceeds the capacity of the egress link, the proposed design guarantees minimal bandwidth by limiting output traffic and priority forwarding. Compared with approaches based on OpenFlow, our design on programmable

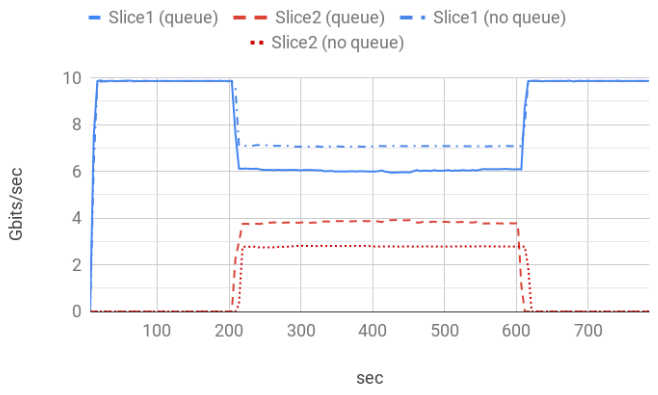

Fig. 11. Queue Facility Comparison Result

P4 switch is not constrained by limitations such as fixed action and packet processing. We also avoid policer interference problem when applying multiple policies to the same flow.

We have conducted experiments on a production network. The results show that both the limited bandwidth and the guaranteed bandwidth can be effectively ensured. The results also show the impact of priority queue on the effectiveness of bandwidth management.

References

[1] Q. Wang et al. ”Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases.” IEEE Transactions on Broadcasting (2019).

[2] Q. Duan. ”Network-as-a-service in software-defined networks for end-to-end QoS provisioning.” 2014 23rd Wireless and Optical Communication Conference (WOCC). IEEE, 2014.

[3] P. M. Mohan et al. ”Performance study of TCP flows with QoS-supported OpenFlow in data center networks.” 2013 19th IEEE International Conference on Networks (ICON). IEEE, 2013.

[4] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers. ”Providing bandwidth guarantees with OpenFlow.” 2016 Symposium on Communications and Vehicular Technologies (SCVT). IEEE, 2016.

[5] N. Kitsuwan and E. Oki. ”Traffic splitting technique using meter table in software-defined network.” 2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR). IEEE, 2016.

[6] F. Paolucci et al. ”P4 Edge node enabling stateful traffic engineering and cyber security.” Journal of Optical Communications and Networking 11.1 (2019): A84-A95.

[7] N. McKeown et al. ”OpenFlow: enabling innovation in campus networks.” ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[8] P. Bosshart et al. ”P4: Programming protocol-independent packet processors.” ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.

[9] P. Berde et al. ”ONOS: towards an open, distributed SDN OS.” Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.

[10] ONOS Support for P4. url: https://www.opennetworking.org/wp-content/uploads/2018/12/ONOS-support-for-P4.pdf

[11] OpenFlow Switch Specification Version 1.3.4 March 2014. url: https://www.opennetworking.org/wp-content/uploads/2012/10/openflow-switch-v1.3.4.pdf

[12] A Two Rate Three Color Marker. url: https://tools.ietf.org/html/rfc2698

[13] ONOS Basic Pipeline. url: https://github.com/opennetworkinglab/onos/tree/master/pipelines/basic

[14] P4_16 Language Specification Version 1.1.0 November 2018. url: https://p4.org/p4-spec/docs/P4-16-v1.1.0-spec.html

[15] iPerf - The ultimate speed test tool for TCP, UDP and SCTP. url: https://iperf.fr/