A Study on Open vSwitch Queues Toward Realizing a Multi-QoS Campus Network

Takehiro Sato and Naoaki Yamanaka
Keio University
3–14–1 Hiyoshi, Kohoku, Yokohama, Kanagawa, Japan
Email: takehiro.sato@yamanaka.ics.keio.ac.jp

Hunter Werlla, Marco Tacca and Andrea Fumagalli
The University of Texas at Dallas
800 West Campbell Road, Richardson, Texas, USA

Abstract—A network infrastructure that deals with a wide variety of bandwidth requirements has been required to make research activities more valuable. We aim to build such a multi-Quality-of-Service (QoS) campus network by using open-source softwares. In this paper, we study the behavior of the queue function provided by Open vSwitch, which is one of the OpenFlow software switches, and discuss its applicability to the multi-QoS campus network.

I. INTRODUCTION

With the development of large-scale computing technology such as big data processing, the demand for a network infrastructure that provides high bandwidth with a Quality-of-Service (QoS) guarantee has been increasing. This is because the total computational speed can be limited by poor network performance, even if the performance of the computing system itself is very high. On the other hand, there are also demands of network services that requires small bandwidth, such as file transfers, and they will continue to be widely used. Therefore, the next-generation research network infrastructure is required to deal with network traffic with a wide variety of bandwidth requirements.

Software-Defined Networking (SDN) technology has been emerging in recent years and is expected to realize adaptive management of network traffic. OpenFlow [1] is one of the representative technologies of SDN. We are trying to build a multi-QoS campus network that satisfies the above requirement by using open-source SDN softwares. As the first step of this challenge, we study the behavior of the queue function of Open vSwitch [2] and discuss its applicability to building the multi-QoS campus network in this paper.

II. QoS FUNCTIONS IN OPENFLOW

Some of the open-source software switches support specific functions that can be used to guarantee QoS of each flow. For example, Lagopus switch [3] and ofsoftswitch13 [4] support a meter table, which has been introduced in OpenFlow 1.3. The meter enables traffic policing in OpenFlow network. That means, bandwidth of each flow can be limited by discarding excessive data frames at each switch equipment.

Open vSwitch [2] does not support the meter at the present time of writing (version 2.5.0). Instead of the meter, Open vSwitch provides the queue function that is implemented by using an Open vSwitch database. The queue function enables bandwidth guarantee and traffic shaping by setting maximum rate (max_rate) and minimum rate (min_rate) for each flow at each switch equipment. Ryu OpenFlow controller [5] provides REST API to manage Open vSwitch queues easily.

From the aspect of controllability, we focus on the queue function of Open vSwitch to realize QoS support in this paper.

III. EXPERIMENT ON OPEN VSWITCH QUEUES

A. Overview

In this experiment, we measured the transmission rate of flows to figure out the behavior of the Open vSwitch queues. Figures 1 and 2 show the configuration of the experiment network.

Open vSwitch was installed in Switch 1 and 2, and they were controlled from an orchestrator via a Ryu controller. One or two queues were configured in Switch 1, and each queue was allocated to a single flow. We did not configure any queues in Switch 2 in this experiment. Data frames input from one Ethernet port of Switch 2 were just output to the other port without any processing. We used iperf3 [6] to measure the transmission rate between Host 1 and Host 2.

Table I shows the variation of trials that we conducted to figure out the behavior of the queues.

In this experiment, we considered only TCP traffic. The duration time of each flow was 30 seconds. When there were two flows (trial (3) – (5)), flow #1 starts 10 seconds behind flow #0. These two flows were distinguished by port numbers (e.g., 5001 and 5002). The bandwidth of data plane was limited to 100 Mbps by setting the common max_rate. Hierarchical Token Bucket (HTB) [7] was used as a packet scheduling algorithm in each switch.
TABLE I
TRIALS IN THIS EXPERIMENT.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Number of flows</th>
<th>Queue settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>1</td>
<td>Flow #0 (Queue 0): max_rate 50 Mbps</td>
</tr>
<tr>
<td>(2)</td>
<td>1</td>
<td>Flow #0 (Queue 0): min_rate 30 Mbps</td>
</tr>
</tbody>
</table>
| (3)   | 2               | Flow #0 (Queue 0): max_rate 20 Mbps  
Flow #1 (Queue 1): max_rate 20 Mbps |
| (4)   | 2               | Flow #0 (Queue 0): max_rate 20 Mbps  
Flow #1 (Queue 1): min_rate 20 Mbps |
| (5)   | 2               | Flow #0 (Queue 0): min_rate 20 Mbps  
Flow #1 (Queue 1): min_rate [20, 10, 5] Mbps |

B. Results and discussion

Figures 3 – 7 show the results of trial (1) – (5) respectively. From Fig. 3, we can find that a max_rate queue limits the bandwidth of flow #0 to the specified value (50 Mbps). It seems that the max_rate queue limits the average bandwidth, not the instantaneous bandwidth. From Fig. 4, it can be said that a min_rate queue basically tend to use almost all of the available bandwidth (limited to 100 Mbps by the common max_rate setting in this trial).

When there were two flows, the behavior of each flow changed according to the combination of queues. In the case of two max_rate queues (Fig. 5), the bandwidth of both flows were limited to the specified value (20 Mbps on average). However, in the case of one max_rate queue and one min_rate queue (Fig. 6), the bandwidth of the flow of max_rate queue was suppressed by the flow of min_rate queue. The same behavior was observed even when the starting time of each flow was exchanged. This behavior of queues is theoretically correct, but may cause a problem in configuring multiple QoS classes, such as guaranteed class and best-effort class.

When there were two min_rate queues (Fig. 7), the available bandwidth was divided by two flows. The ratio of allocated bandwidth changed according to the setting value of min_rate queues. Figures 8(a) and 8(b) shows additional results when the setting value of queue 1 is 10 Mbps and 5 Mbps, respectively. In every cases, the minimum bandwidth of each flow was guaranteed by min_rate queue settings.

IV. CONCLUSION

In this paper, the behavior of the queue function provided by Open vSwitch was evaluated in an effort to realize a multi-QoS campus network. The experimental result pointed out that there may be a problem when a max_rate queue and a min_rate
queue are set in the same switch simultaneously. As a future work, we will study how to achieve multiple QoS classes by using these two types of queues. We also have to clarify the behavior of queues when UDP flows exists.

REFERENCES