Adaptive Control Channel Traffic Shaping for Virtualized SDN in Clouds

Yeonho Yoo*, Gyeongsik Yang*, Minkoo Kang, Chuck Yoo
Department of Computer Science and Engineering, Korea University
{*yhyoo, ksyang, mkkang, chuckyoo*}*@os.korea.ac.kr*

***Abstract*—As the number of tenants in clouds grows, virtualized SDN faces a challenge of control channel fairness as the control channels interfere with each other. This paper proposes an adaptive traffic shaping scheme for control channels called "Sincon." Through experiments, Sincon achieves traffic shaping for control channels and improves the variances of control channel throughputs and forwarding setup times up to 3.8 and 2.86 times, respectively.**

***Keywords*-network virtualization; control channel traffic; traffic shaping; SDN; cloud network;**

I. Introduction

Cloud datacenters require the isolation of computing resources between tenants, such as CPU, memory, and network. The isolation for the CPU and memory has been well-understood. For network isolation, the cloud data centers use network virtualization (NV). However, NV does not provide programmability to tenants [1], and there have been efforts called software-defined networking with NV (SDN-NV) that allow each tenant to constitute and program its virtual network (VN) flexibly. In SDN-NV, the network hypervisor (NH) plays a key role in providing an abstraction of network switches (virtual switch) to the tenants. Each tenant uses its own network controller (tenant controller) to control the virtual switches provided by the NH. The tenant controller can install the flow rules for packet forwarding or firewall and monitor the network traffic from these virtual switches. The operations such as flow rule installation or network monitoring belong to the control plane, and their messages (called control messages) are delivered through a control channel. A control channel is a logical path between the tenant controller and a virtual switch, so the total number of control channels is the number of virtual switches. A control channel is implemented as a dedicated network connection.

Because the control channels share the physical network, the throughput of each control channel is critical to the performance of SDN-NV. However, this paper reports that the control channel has a serious fairness problem among control channels – when the number of tenants increases from one to four, the time for flow rule installation (forwarding setup) increases by up to 3.6 times. We find that this is because the control channels interfere with each other. Furthermore, the time of the forwarding setup delays the packet transmission in the data plane, which means that the control channel throughput deteriorates the performance of SDN-NV. Thus far, the importance of control channel fairness has been addressed only in the SDN context, but it has not been investigated in SDN-NV.

In this paper, we propose Sincon, an adaptive control channel shaping scheme for SDN-NV. This scheme uses a traffic shaping mechanism that limits the maximum throughput of a control channel, considering the fairness of control channels. Sincon also adds a queue for the channel to avoid packet drops, and this makes Sincon more effective than existing traffic limiting or metering [2] that simply drops packets. Sincon equips a control channel with a throughput limit. When the usage approaches the throughput limit, Sincon adaptively increases the throughput limit (§III). We implement Sincon in OpenVirteX (OVX) with OpenFlow 1.3 [3] and measure control channel throughputs and forwarding setup time improvements. Overall, Sincon improves the fairness of control channel throughput by up to 3.8 times. Thus, the forwarding setup time decreases by up to 2.9 times.

II. Background and Motivation

A. SDN-NV

Fig. 1 shows the SDN-NV architecture. Each tenant has a tenant controller that is an SDN controller, such as ONOS, POX, and OpenDayLight. The tenant controller can define its desired VN topology. The NH in Fig. 1 creates virtual switches and manages all the network switches in the physical network. It also establishes the control channel between a virtual switch and the tenant controller. In datacenters, the NH and tenant controllers typically run in different physical servers. For example, in Fig. 1, the controllers run in server A and B, and the NH runs in server C.

Up to now, a variety of NHs have been proposed (e.g., FlowVisor [4], OpenVirteX [5], and Libera [1]). Each NH has its own goal, such as network slicing [4], address virtualization [5], and scalable and flexible framework [1]. However, the control channels have not been investigated for

*The first two authors contributed equally to this paper.
†Corresponding author: Chuck Yoo.
‡This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (No. 2015-0-00280, (SW Starlab) Next generation cloud infra-software toward the guarantee of performance and security SLA) and by Next Generation Engineering Researcher Program of National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (No. NRF-2019H1D8A2105513).

2159-6190/20/$31.00 ©2020 IEEE
DOI 10.1109/CLOUD49709.2020.00013

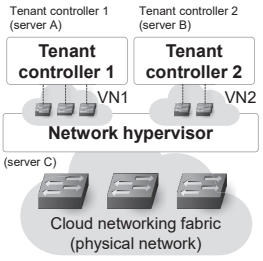

Figure 1: SDN-NV architecture

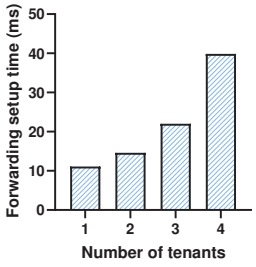

Figure 2: Average forwarding setup time

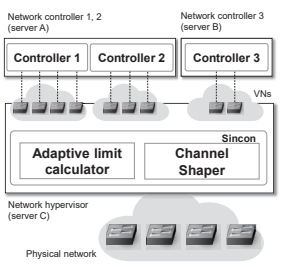

Figure 3: Sincon components

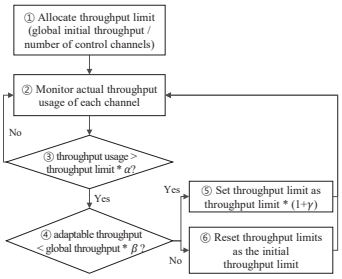

Figure 4: Adaptive limit calculator flowchart

fairness. We illustrate the fairness problem with experiments.

B. Control Channel Throughput Interference

For experiments, we create a 4-ary fat-tree topology as a physical network and a VN with 10 switches within the physical network. We increase the number of tenants from 1 to 4, and each additional tenant increases the number of control channels by 10. After each VN is created, 20 TCP connections have the same workload in the VN, and the control messages are evenly sent to virtual switches. We measure the time for the forwarding setup in OVX. Fig. 2 shows the experiment results. The forwarding setup time increases by 3.6 times from one tenant to four tenants. To understand this delay, we measure the throughput usage of each control channel. We find that the maximum throughput of a control channel is 242% of the minimum throughput of other control channels, among four tenants. In other words, some control channels are much busier than other control channels. With further analysis, we find that control channels that send messages earlier than other control channels dominate the network bandwidth.

In previous studies, DevoFlow [6] merges the multiple flow rules into a smaller number of flow rules. Thus, the traffic for flow rule installation decreases. In SDN-NV, LiteVisor [3] also reduced the control channel traffic for forwarding setup by aggregating the flow rules using the location of the server and switches. However, the interference of control channels is a different problem from reducing control channel traffic. To overcome this interference problem, Sincon introduces an adaptive throughput shaping technique for the control channels, which is explained in Section III.

III. Sincon Design

We design Sincon as a part of the NH. Fig. 3 shows the Sincon components, which include 1) the adaptive limit calculator and 2) the channel shaper. In Sincon, a control channel is a TCP connection with a queue. The queue stores the packets of the channel. For each control channel, Sincon has two operations: ”channel_read” and ”channel_write.” Channel_read processes the packets from the tenant controllers, and channel_write sends the packets to the tenant controllers. The adaptive limit calculator sets the throughput limit of each control channel in accordance with the actual throughput usage (§III-A). In addition, the channel shaper performs traffic shaping according to the throughput limit from the adaptive limit calculator. (§III-B).

A. Adaptive Limit Calculator

The adaptive limit calculator determines the throughput limits. A throughput limit is the amount of throughput that each control channel can use, and the channel shaper performs traffic shaping according to this value.

Fig. 4 shows a flowchart of the adaptive limit calculator. The adaptive limit calculator works as follows. First, it checks the available throughput of a physical network link between the two servers running the NH and controllers. We call this throughput as the “global throughput.” Then, based on the global throughput, the initial throughput limit of each control channel is determined (① in Fig. 4). Suppose that the servers A and C in Fig. 3 are connected through a 1,400 Mbps link. Sincon uses a portion of the 1,400 Mbps, called the ”global initial throughput,” for the initial throughput limit. The remaining throughput of the global throughput is called ”adjustable throughput.” Sincon sets the initial throughput limit of each channel fairly when the VN is started. Sincon divides the global throughput into 1:1 and makes the global initial throughput and adjustable throughput become 700 Mbps. The adaptive limit calculator then sets the initial throughput limit of each channel (global initial throughput/the number of channels). In the example in Fig. 3, the number of control channels is 7 and, thus, each control channel’s initial throughput limit becomes 100 Mbps. Second, while monitoring the throughput usage, the adaptive limit calculator increases the throughput limit of a control channel when it fully utilizes the initial throughput limit. This is to cover the actual throughput usage of each control channel that differs depending on the network status. For example, with a virtual switch acting as an edge switch[1], the tenant controllers install more flow rules than for the other switches, such as for firewall or packet encapsulation in datacenters. Thus, the amount of control traffic on the edge switch can be two times higher than in the other switches [3]. In this situation, the adjustable throughput is used to provide more throughput limit to the control channels that fully utilize the initial throughput limit.

[1]We refer to a switch attached to a host as an edge switch.

So, the adaptive limit calculator checks the actual throughput usage of each control channel whenever a certain number of control channel messages arrive[2] (②). If the actual throughput usage approaches the given limit value, for example, at 95% (α) of the throughput limit (③), the adaptive limit calculator increases the throughput limit (γ, ⑤). The increased throughput limit comes from the adjustable limit. When the throughput limit of a control channel increases, the adjustable throughput reduces by that amount. When increasing the throughput limit of a control channel, the adaptive limit calculator checks the remaining adjustable limit. If the remaining adjustable limit is less than a certain amount, say 70 Mbps, which is 5% (β) of the global throughput (④), the adaptive limit calculator resets the throughput limits of all the control channels back to be the global initial throughput limit. (⑥).

B. Channel Shaper

The channel shaper performs traffic shaping to meet the throughput limit by providing a delay between control channel processing. First, the channel shaper handles the channel_read and channel_write operations. When the channel shaper processes the packets from/to a control channel, it calculates the throughput processed. The channel shaper then sets a delay to the next processing following the throughput limit. When the throughput limit is almost utilized, the delay becomes higher in proportion to the throughput. For example, if the throughput limit is 100 Mbps, and the current throughput is 95 Mbps (which is close to the throughput limit), the channel shaper gives a higher delay (we omit the details due to the page length). Note that because each control channel has a queue, the packets for the control channel are not dropped, even the delay is given. In addition, when the number of processed packets is small (e.g., 30 Mbps usage when the throughput limit is 100 Mbps), the channel shaper sets a shorter delay, so the channel_read and channel_write occurs earlier. In this manner, the throughput of each control channel is managed to be close to the throughput limit.

IV. Implementation and Evaluation

We implement Sincon in the OpenFlow 1.3 based OVX. Two metrics are evaluated: 1) control channel throughput and 2) forwarding setup time. For each metric, the number of tenants varies from one to four. The experiment topology and traffic generation are similar to the one used in §II-B. We compare the evaluation results with the OVX, which does not have Sincon. Each evaluation result is measured more than 20 times to obtain a reliable result.

To view the fairness among control channels, Fig. 5 shows the standard deviation of the control channel throughputs for channel_read and channel_write operations when the number of tenants is four (i.e., 40 virtual switches). According to these results, the throughput variance improves by up to 3.8 times, indicating that the interference between control channels is reduced with Sincon.

[2]We set the number of control messages to trigger the adaptive channel monitor at 5. However, this value can be changed depending on the network.

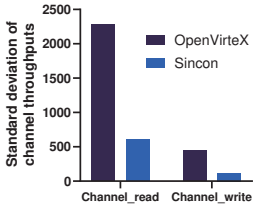

Figure 5: Control channel throughput

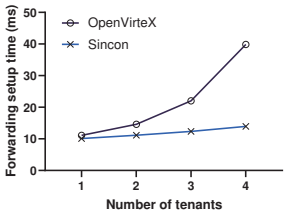

Figure 6: Average forwarding setup time

In addition, Fig. 6 illustrates the forwarding setup time that improves from the fairness among control channels. The results of OVX show that the control channel traffic interferes with each other as the number of control channels increases. With Sincon, the forwarding setup time mostly stays a constant time independent of the number of tenants. The maximum improvement is up to 2.9 times when the number of tenants is four. These results indicate that Sincon improves the fairness of control channels.

V. Conclusion and Future Work

We present Sincon that provides an adaptive control channel traffic shaping scheme. Sincon introduces the adaptive limit calculator and the channel shaper. With its implementation, Sincon demonstrates improvements with the standard deviations of the control channel throughputs and forwarding setup times of up to 3.8 and 2.9 times, respectively.

In the future, we plan to design an algorithm for predicting the throughput of each control channel in advance, to provide a more stable and fairer virtual network system.

References

[1] G. Yang, B.-y. Yu, H. Jin, and C. Yoo, "Libera for programmable network virtualization," *IEEE Communications Magazine*, vol. 58, no. 4, pp. 38–44, 2020.

[2] "Comparing traffic policing and traffic shaping for bandwidth limiting," Oct 2019. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

[3] G. Yang, B.-Y. Yu, S.-M. Kim, and C. Yoo, "LiteVisor: A network hypervisor to support flow aggregation and seamless network reconfiguration for VM migration in virtualized software-defined networks," *IEEE Access*, vol. 6, pp. 65 945–65 959, 2018.

[4] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *OSDI*, vol. 10, 2010, pp. 1–6.

[5] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: Make your virtual SDNs programmable," in *The third workshop on Hot topics in software defined networking*, 2014, pp. 25–30.

[6] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *the ACM SIGCOMM 2011 conference*, 2011, pp. 254–265.