Case Western Reserve University

Final Project Report

Priority Based Virtual Network Bandwidth Guarantee Methods in Software Defined Network

---

Jiaqi Yang

Ziqin Zhai

December 13, 2019

Date	Version Number	Note
December 12	V 1.0	Initial Release Documentation
December 13	V 1.1	Team Reviewed and Signoff

1. Abstract

With the development of various network services, the load of network links is increasing. As a result, the rational distribution of existing network resources has become an urgent problem. Along with the introduction of the OpenFlow protocol, the emergence of the Software-Defined Network (SDN) provides a promising approach to address the bandwidth-guarantee problem. In this paper, the bandwidth-guarantee strategy is written in the Ryu controller in order to allocate the bandwidth resource. In both proposed methods, we invoke the Hierarchical Token Bucket (HTB) algorithm to create queues, assign priorities, and maintain lower and upper limits of queue bandwidth. Finally, in the experiment section, the bandwidth-guarantee methods are implemented in the virtual machine and test to show the effectiveness of bandwidth allocation.

Keywords - Bandwidth Guarantee; Software Defined Network; Ryu; SDN; OpenvSwitch; HTB

2. Introduction

With the rapid development of the Internet, the Internet took off faster than anyone could have predicted. Nowadays, every device ranging from wearables, home appliances, and smartphones to cars, security systems, and city management systems is connected to the Internet. In a word, all kinds of services and applications based on the network are growing exponentially. The rapid development of the Internet also brings new challenges. Along with the immense growth of devices connecting to the network, the network users also have high expectations regarding good performance: instead of only sending emails and reading forums, the user behaviors on the

Internet changes dramatically to large online gaming and high-resolution video streaming, which both require smaller latency and larger bandwidth compared to ten years ago. In past decades, the research to address this crucial Quality-of-Service (QoS) problem point to the strategy of bandwidth allocation, which is proven to optimize network state and improve users' experiences. Most studies are mainly confined to the traditional network and based on specific services to guarantee bandwidth allocation. For example, one of these services protect the video from distortion or mosaic in the multi-service [1]; however, when the user's bandwidth is not guaranteed due to network congestion, the service is unable to function due to the fundamental problem of bandwidth limitation. As a result, the only way to solve this QoS problem is to guarantee total bandwidth which could further support all types of services run smoothly.

The introduction of the Software-Defined Network (SDN) is considered a promising technology for facilitating the bandwidth-guarantee process. SDN is a network paradigm that separates the control plane and data planes. The centralized-SDN controller communicates with the switches like Open vSwitch [2] via a Southbound interface like OpenFlow [3]. Therefore it provides the capabilities to realize the flexible control of traffic and the rapid deployment due to control and data plane decoupling. In this paper, we propose two HTB-based approaches that combine the SDN controller Ryu [4], OpenFlow, and Open vSwitch to solve the bandwidth guarantee problems.

The paper is organized in the following manner. Section 3 gives a brief overview of related works that propose similar HTB-based technology to address bandwidth-guarantee problems. Section 4 gives an introduction to the system model and the concerned components in our design including Ryu, OpenFlow, Open vSwitch, and its HTB queueing technique. Section 5 presents

the proposed bandwidth-guarantee methods: the naïve HTB-approach and the more sophisticated multipath HTB-approach. The methods calculate bandwidth for each device using individual priority and the total bandwidth available. Section 6 briefly describes the network topology, the methodology of experimental design, and the test results in order to show the validity and effectiveness of the proposed approaches. In Section 7, we conclude our research and talk about the potential future work developed from this research.

3. Related Works

Since the introduction of OpenFlow [3] in 2011, the research in the field of Software-Defined Network (SDN) increases dramatically. Among them, some of the researches proposed similar approaches utilizing HTB-based traffic control schemes. [11] In short, the hierarchical token bucket (HTB) is used in Open vSwitch and allows each traffic class to be configured with guaranteed bandwidth for packet transmission. [5]

In the paper by Qu, Liao, An, and Fan [6], the HTB technique is used to provide guaranteed bandwidth for Virtual Networks (VN) using Open vSwitch. This paper provides the basic idea of Jiaqi's naïve HTB-approach: the bandwidth allocated to each virtual network is directly proportional to its priority, and therefore the VN with higher priority is granted with more guaranteed bandwidth. Similarly, the proposed approach by Wang and Shi [1] monitors the real-time bandwidth usage with Floodlight [8] and dynamically configures HTB to reallocate VN bandwidth to improve network resource usage based on the monitored bandwidth usage.

Moreover, Lee and Kim [7] take a step further: they modify and optimize the HTB kernel module in Linux IP/MPLS router and prioritize the traffic flows into real-time, non-real-time,

and best-effort classes based on the timing requirement and dynamically adjusts the bandwidth assignment of classes in accordance to the remaining bandwidth amount inside the HTB buckets. Meanwhile, Bhattacharjee, Gopal, Ngangoua, and Raghunath's work on their HTB-based approach TrafficLight [8] is more ambitious: it provides a management system to facilitate users to configure the network and eventually realize self-learning and dynamic adaptability on network resources management and bandwidth guarantee.

4. System Model and Components

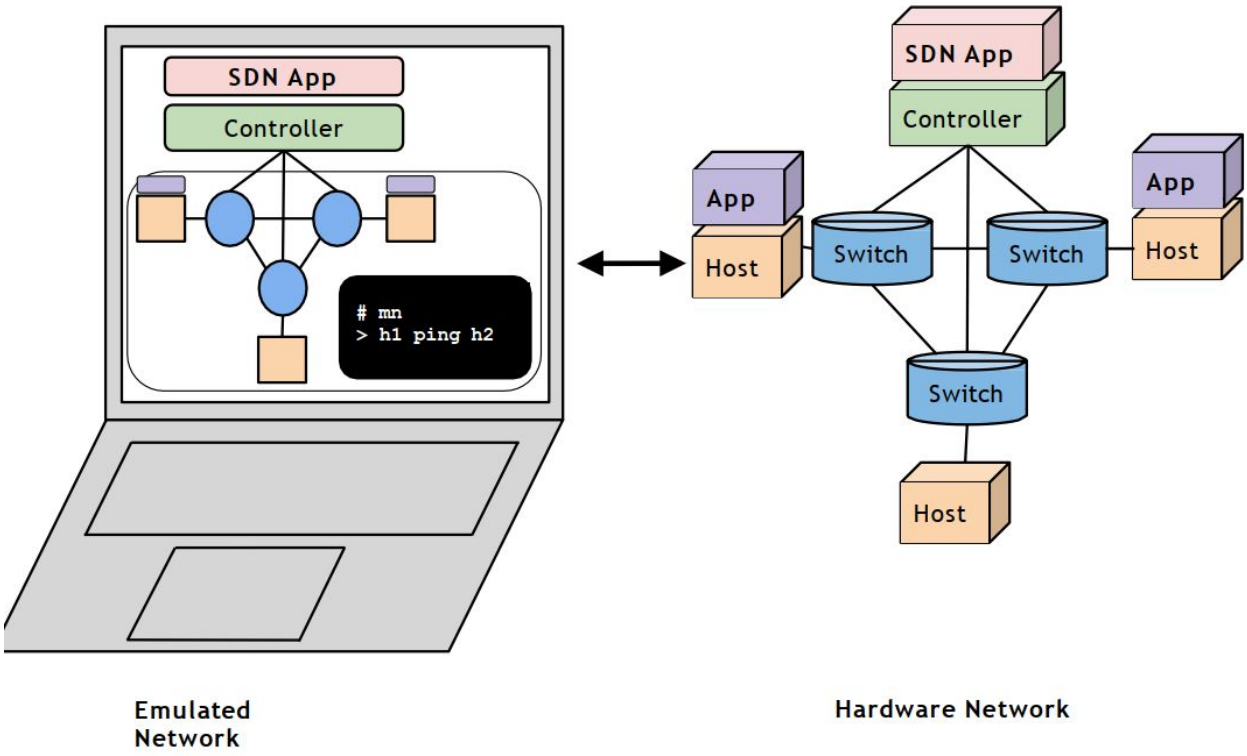

Figure 1: System Model [10]

4.1 Ryu and OpenFlow

Ryu [4] is a component-based software-defined networking framework. Ryu provides software components with well-defined API that makes it easy for developers to create new network management and control applications. Ryu supports various protocols for managing the network, such as OpenFlow. In our approaches, we utilize OpenFlow version 1.3, which is fully supported by Ryu. OpenFlow version 1.3 was released in 2012. And it supports more features than version 1.0 such as VLANs, logic ports, tunneling and per-flow traffic meters which means this version has the ability to handle both the VN and the QoS.

4.2 Open vSwitch and HTB

Open vSwitch [2] is a production quality, multilayer virtual switch licensed under the open-source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag).

The hierarchical token bucket (HTB) [5] has been implemented for traffic control in Linux as the replacement of the Class-Based Queuing (CBQ) technique, and later it is adapted in many software switches and routers, such as Open vSwitch. The HTB applies a class-based hierarchical system for system control where traffic classes, each configured with guaranteed bandwidth, are implemented in a tree structure.[11] To utilize HTB, every queue of the switch's port should be configured with a lower bound, which represents the minimum bandwidth for the queue, and a higher bound of available bandwidth, which represents the upper limit of bandwidth allocation for the queue. [6] Therefore, HTB maintains the minimum bandwidth for each queue

to guarantee transmission when the traffic is overloaded and automatically allocates the spare bandwidth proportionally to the queues based on each queue's priority in order to maximize the utilization of the total bandwidth and therefore maintain the bandwidth usage close to 100% at all times.

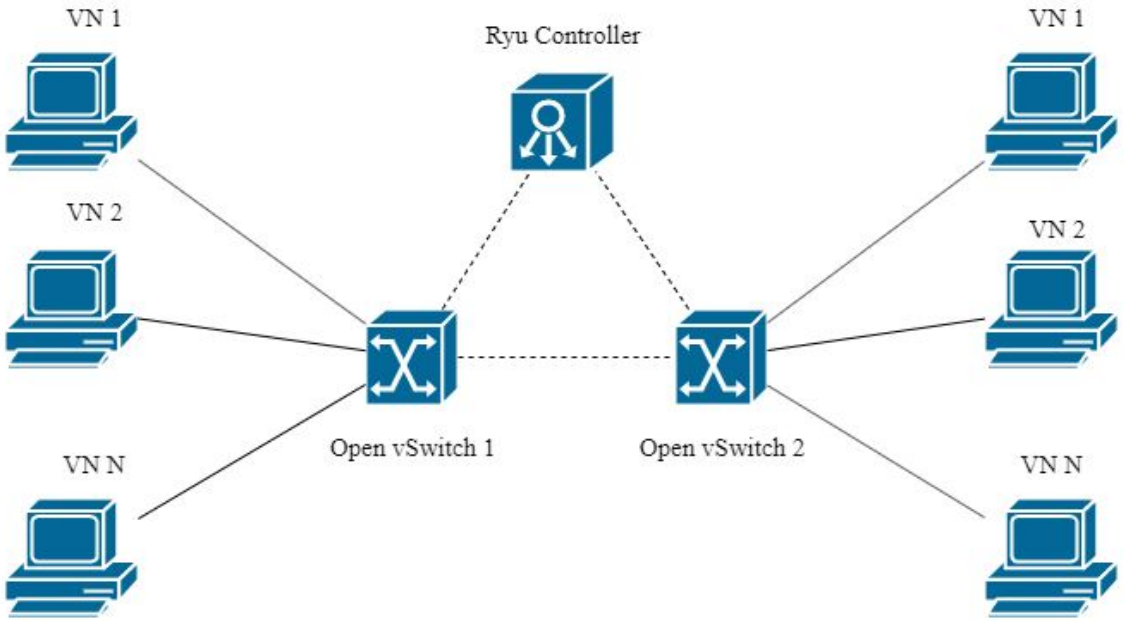

Figure 2: System Model and Network Topology

4.3 System Model

Figure 2 shows the typical SDN framework which contains both the control plane, Ryu controller and the data plane, two Open vSwitch switches. The network topology shows two virtual switches are connected to the Ryu controller, while one or more devices are connected to each virtual switch. Assume the total bandwidth between two virtual switches is constant, all traffic will compete for bandwidth without QoS if all the VNs are overloaded. In this scenario,

the bandwidth-guarantee method is crucial to ensure that each VN has a minimum bandwidth and the total bandwidth utility should be close to 100% at all times.

5. Proposed Bandwidth-Guarantee Method

5.1 Naïve Method

In this section, we propose a bandwidth guarantee method based on priority. The theory work is based on the study by Qu, Liao, An, and Fan's study [6], while the implementation details are not provided in their work. The implementation part could be divided into two parts: priority calculation and queue configuration as well as Ryu controller application.

The first step is to set the queue by priority. As demonstrated in Figure 2 above, the network topology used in the study consists of two Open vSwitch virtual switches s1 and s2, each connected with one or more VNs h1, h2, and so on. Each VN has a priority value Pr assigned to it: if a VN has a higher priority, the VN would be allocated with more bandwidth. To use the HTB algorithm, each queue should be assigned a lower and an upper limit of bandwidth.

If the priority of each VN is represented by Pr1, Pr2, …, Pr_N, the summation of the priority can be got from (1)

$$Pr\_sum = \sum_{i=1}^{N} Pr_i \tag{1}$$

We could easily calculate the summation of priorities Pr_{sum}. If the lower limit of bandwidth allocation for each VN is represented by B1, B2, …, B_N, we could calculate based on the total bandwidth available B_ttl and the priorities of each VN from (2)

$$B_i = B\_ttl \times \frac{Pr_i}{Pr_{Sum}} \quad (2)$$

After the calculation, we configure the queue on the port on the port of virtual switch with the lower bound of the bandwidth as B_N and the upper bound of the bandwidth B_ttl as following:

ovs-vsctl -- set Port eth0 qos = @newqos

-- -- id = @newqos create QoS type = linux-htb other-config:max-rate = B_ttl queues = 1 = @q1, 2 = @q2, ..., N = @q_N

-- -- id = @q_0 create queue other-config: min-rate = B_0 other-config: max-rate B_ttl

-- -- id = @q_1 create queue other-config: min-rate = B_1 other-config: max-rate B_ttl

… …

-- -- id = @q_N create queue other-config: min-rate = B_N other-config: max-rate B_ttl

The command above creates N queues on the port eth0, each has a lower bound and a higher bound of available bandwidth.

The second step is to write the Ryu controller application. In the application, the most important part is the _packet_in_handler function, which decides the actions to a specific packet feature in order to achieve bandwidth guarantee. The logic is demonstrated as following:

1) Get the in_port, eth_dst, and eth_src of the packet from the msg.match module of Ryu.
2) Learn the relationship of the eth_src and the in_port in VN's table to avoid flood next time

3) If the eth_dst is not in the table, then make a packet-out message to the switch in order to flood the packet in the VN.
4) If the eth_dst is in the table, then make a set_queue to the switch to install a flow table to forward the matched packets to the corresponding queue.

While the first three steps are basically learning switch implemented in assignment 1, the last step is how our Ryu application guarantees the bandwidth.

5.2 Multipath Method

In this section, we proposed a new method called Multipath. Multipath can be implemented by select group in Ryu and it is easy to be implemented. Before we implement Multipath, we use OVS command to set up the queues since we cannot create queues with OpenFlow, we also set up the max-rate and min-rate with OVS at the same time. The main idea to implement Multipath is utilizing the group_table in Ryu controller. The following steps are required for implementation:

1) Every port and every queue which matches the port should be defined. To set queue in each port, we use the function OFPActionSetQueue() in OpenFlow1.3
2) In OpenFlow, each enqueue operation has only one queue_id. As a result, specifying a certain port_id is necessary for a certain enqueue action. For example, to put queue2 out of port2, set an action: action_2 = [queue_2, ofp_paser.OPFActionOutput(port_2)]
3) Set the weight of each port.
4) Set buckets to provide an action list for every flow matched.

5) Define the group's id (set group_id = 50) and set the request

6. Experimental Validations

We set the same topology to test these two methods. By testing the bandwidth between host1 and host2 and the bandwidth between host1 and host3, we can compare these two methods by analyzing the results of the load balancing.

Before we start our experiment, we use OVS command to set the queues. For the first method, We put Queue 0 on s3_eth2 and set the max-rate = 250000000. We put Queue 0 and Queue 1 on s3_eth3 and set the max-rate = 300000000:

ovs-vsctl -- set Port s3_eth2 qos = @newqos

-- -- id = @newqos create QoS type = linux-htb other-config:max-rate = 250000000 queues = 0 = @q0 -- -- id = id = @q0 create Queue other-config:min-rate = 8000000 other-config:max-rate = 150000000

ovs-vsctl -- set Port s3_eth3 qos = @defaultqos

-- -- id = @defaultqos create QoS type = linux-htb other-config:max-rate = 300000000 queues = 1 = @q1 -- -- id = id = @q1 create Queue other-config:min-rate = 5000000 other-config:max-rate = 200000000

By test the first method, we get:

Iperf: testing bandwidth between h1 and h2

Results: ['162 Mbits/sec', '180Mbits/sec']

And

Iperf: testing bandwidth between h1 and h3

Results: [‘147 Mbits/sec’, ‘173Mbits/sec’]

According to our result, we get that the max bandwidth between host 1 and host 2 is 162Mbits/sec, the max bandwidth between host 1 and host 3 is 147 Mbits/sec. As a result we can see that the flow from host 1 to host 2 choose Queue 0 on s3_eth2, and the flow from host 1 to host 3 choose Queue 1 on s3_eth3.

By testing the second method, we get:

Iperf: testing bandwidth between h1 and h2

Results: [‘299 Mbits/sec’, ‘312Mbits/sec’]

And

Iperf: testing bandwidth between h1 and h3

Results: [‘149 Mbits/sec’, ‘161Mbits/sec’]

According to our result, we get that the max bandwidth between host 1 and host 2 is 299Mbits/sec, the max bandwidth between host 1 and host 3 is 149 Mbits/sec. As a result we can see that the flow from host 1 to host 2 choose Queue 0 on s3_eth3, and the flow from host 1 to host 3 choose Queue 0 on s3_eth3.

Comparing the results we can clearly see that by using Multipath, the usage of the bandwidth is more efficient than the first method.

7. Conclusion and Future Works

In this paper, we explore two bandwidth-guarantee methods. In the method, we introduce the approach to configure the QoS queues on the virtual switch port and develop two applications deployed on the Ryu controller. The experiment results demonstrate that our bandwidth-guarantee methods show good performance for the VNs with different priorities and maintain a high total bandwidth utilization rate at all times. In our future study, we will test the method in a more complicated network topology setup like a large-scale data center and deploy in the physical switches and servers to further provide our approaches' effectiveness and validity.

8. Workload Division

We collaboratively work on the code and report. Both team members work on all the programming parts including naive algorithm and multipath algorithm, as well as report writing by dividing it equally.

Reference

1. J. Wang and P. Shi, "Research on Bandwidth Control Technology Based on SDN," *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, Xi'an, 2018, pp. 705-708. doi: 10.1109/IMCEC.2018.8469269

2. "Open vSwitch", [online] Available: http://openvswitch.org/.

3. N. Mckeown et al., "OpenFlow: Enabling Innovation in Campus Network", ACM SIGCOMM, 2008.

4. "Ryu", [online] Available: https://github.com/osrg/ryu/.

5. M. Devera, "HTB Linux queuing discipline manual - user guide", [online] Available: http://luxik.cdi.cz/~devik/qos/htb/manual/userg.html.

6. Q. Fu, L. Qing, A. Yingzhu, F. Yamei, "A priority-based virtual network bandwidth guarantee method in software-defined network", Proc. 6th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS), pp. 153-156, 2015.

7. C.-H. Lee, Y.-T. Kim, "QoS-aware hierarchical token bucket (QHTB) queuing disciplines for QoS-guaranteed DiffServ provisioning with optimized bandwidth utilization and priority-based preemption", Proc. Int. Conf. Inf. Netw. (ICOIN), pp. 351-358, 2013.

8. "Floodlight", [online] Available: http://www.projectfloodlight.org/floodlight/.

9. T. Bhattacharjee, V. Gopal, L. N. Nganggoua, C. Raghunanth, TrafficLight: Network Traffic Monitoring and Allocation, May 2018, [online] Available: http://people.cs.vt.edu/~tirtha23/CSGrad/TrafficLight.pdf

10. A. Wang, 'Computer Communication Network II lec-3', Case Western Reserve University, 2019

11. S. S. W. Lee and K. Chan, "A Traffic Meter Based on a Multicolor Marker for Bandwidth Guarantee and Priority Differentiation in SDN Virtual Networks," in IEEE Transactions on Network and Service Management, vol. 16, no. 3, pp. 1046-1058, Sept. 2019. doi: 10.1109/TNSM.2019.2923110