고확장성 SDN을 위한 랭킹 기반 플로우 교체 기법

뉘엔 트리 투안 힙*, 김경백*
*전남대학교 전자컴퓨터공학부
e-mail: tuanhiep1232@gmail.com, kyungbaekkim@jnu.ac.kr

Ranking-based Flow Replacement Method for Highly Scalable SDN

Hiep T. Nguyen Tri*, Kyungbaek Kim*
*Dept of Electronics and Computer Engineering, Chonnam National University

요 약

Software Defined Network (SDN) separates control plane and data plane to achieve benefits such as centralized management, centralized provisioning, lower device cost and more flexibility. In SDN, scalability is an important issue. Centralized controller can be a bottle neck and many research tried to solve this issue on the control plan. However, scalability issue does not only happen in the control plane, but also happen in the data plane. In the data plane, flow table is an important component and its size is limited. In a large network operated by SDN technology, the performance of the network can be highly degraded because of the size limitation of a flow table. In this paper, we propose a ranking-based flow replacement method, Flow Table Management (FTM), to overcome this problem.

1. Introduction

In the past few years, SDN (Software Defined Networking) has become a hot topic in computer networking. In SDN, dummy devices in data plane are controlled by a centralized controller in control plane. Whenever a device receives a packet, it matches the packet header with flow entries which are stored in a flow table inside the device. If there is no flow entry is found, the device will send a request to ask the controller how to handle the packet. Then, the controller might install flow entries to the device. Because of separating control plane and data plane, SDN offers some benefits for the network operator such as centralized management, centralized provisioning, lower device cost and more flexibility. With these benefits, SDN is promised as a valuable solution for a back bone network which contains a lot of devices.

However, SDN also issues new problems such as scalability. In a large network, a centralized controller has to manage thousands of devices and conduct millions of requests per second. The centralized controller can become a bottle neck that decreases the network performance significantly. Past research has tried to handle this issue [1] [2]. The concept of distributed controllers is one of solutions, in which each controller controls a part of a large network that has less number of devices and less number of requests.

The scalability issue does not only happen in the control plane but also happen in data plane. Generally many SDN devices employ Ternary Content-Addressable Memory (TCAM) as storing the flow table and lookup flow entries for a given packet header. TCAM is a specialized high-speed memory that searches its entire contents in a single clock cycle, but it has some disadvantages; such as high power consumption, high cost, and low utility of ASCI space. Increasing flow table size can lead to other issues such as price, energy and size of a device. Therefore, the flow table size of a network device is limited. In a large network which contains thousands of network devices and millions of end user devices, the number of flow entries might be up to millions. This is a huge number if we compare with the ability of managing flow entries in a network device. For example, a 5406zl switch can support about 1500 OpenFlow rules or 64000 forwarding entries for standard Ethernet switching [3]. Hence, a flow table of a device can be easily filled with enormous number of flows.

When a flow table is full, for every packet which doesn't match any flow entry in the flow table and the network device has to send a request to the controller. If the controller fails to install a new flow entry to the device, an option for the controller is to replace an old flow entry by the new flow entry and let the network device does its function. Another option is that the controller forwards packets by itself. That is, the controller might send packets to appropriate network devices which might be the closest network device to the destination of a packet. In the second option, the controller might have to process a huge number of packets if the packet belongs to a high loaded flow. Moreover, it may also lead to other problems such as bypassing network security functions. For example, when we put a firewall to prevent a malicious traffic from reaching a destination and the controller handles this traffic flow, the traffic can bypass the firewall. Like the first case the replacement of flow entries can be used for preventing these possible issues with the second option. But the controller might spend a lot of resource to replace flow entries. If the number of active flows is larger than flow table size, the controller has to repeatedly replace flow entries.

In our work we focus on the scalability issue in the data plane. A past approach was proposed to solve this issue is to combine flow entries and reduce the number of flow entry [4]. In fact, combining flow entries contains hidden issues

because the combined flow entry comprises equal or more than the original flow entries. When the controller fails to install flow entries because flow table is full, as we discuss, the controller should let packet goes through network devices as it always does. Hence, a replacement strategy is required to improve the performance. For example, we could replace flow entries that are for handling lower load traffic by new flow entries that are for handling high load traffic. In order to do that, we have to collect statistic information from the network device. However, if the number of active flows is larger than flow table size, the replacement can be repeatedly happened. Therefore, the controller should consider whether a new flow entry should be replaced or not. It might make the controller process a lot of packets that match non-installed flow entry. A cache module which can quickly handle the packets that match non-installed flow entry is an answer for this issue. In this paper, we propose a control plane flow management which replace and cache flow entries smartly to handle scalability issue in control plane. We describe the design of the required modules of the management system.

2. Related Works

2.1 SDN Operation

In this section, we explain the basic SDN operations. The architecture of SDN is composed of three layers such as Application layer, Control layer, and Infrastructure layer [5]. The infrastructure layer (or the data plane) includes network element such as switches and access points. The role of network element is to forward a packet to its destination or drop it. The control layer includes a controller which interacts with the network elements on the infrastructure layer through southbound interface. It also provides an interface to applications of the network (northbound interface). The application layer includes applications which control the network via northbound interface based on the policies given by network operator.

Match fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 1. Flow Entry in a Flow Table

Generally, the northbound interface is implemented as a RESTful API, and the southbound interface implements the OpenFlow Protocol [6]. The OpenFlow Protocol is a protocol for a controller to communicate with network devices for configuring and monitoring them. In the OpenFlow Protocol, each network device has a flow table which contains flow entries. Figure 1 describes fields of a flow entry in a flow table. The *match fields* consist of the information of ingress ports and packet header information, such as source/ destination IP/MAC address, Ethernet type and source/ destination port, and other metadata which are specified in flow tables of other network devices. When a network device receives a packet, it looks up its flow table and tries to find flow entries which have the matched information of the packet header. If the packet header matches multiple flow entries, the flow entry which has highest *priority* value is used. The *counters* field indicates how many times the flow entry is used by increasing its value whenever it is used. The *instruction* field has a list of actions which indicate how to handle packets such as forwarding packets to output ports, dropping packets, and modifying the payload or the header of the packets. The *timeouts* field is used for cleaning up flow entries, that is, deleting flow entries from a flow table. The *cookie* field can be used by controller to filter flow statistic, flow modification and flow detection. This cookie field is not used when processing packets.

Figure 2 illustrates the basic SDN operations for transferring traffic of a flow. Firstly, a source host sends the first packet of a flow to a network device (1). Secondly, the network device looks up matched flow entries. If there is no matched flow entry, the network device sends a request to the controller (2). In the controller, the request will be forwarded to the control application. After getting the request, based on the knowledge of the network, the application makes a decision and tries to install flow entries to all of the network devices along the path of the packet (3). After installing the flow entries, the application sends back the packet to the network device which requests the flow entries as well as to the last network device of the path of the packet. Then the last network device forwards the first packet to the destination host (4). After deploying flow entries in every network devices related to the flow, the network devices handle the following packets of the flow by referring the instruction field of the deployed flow entry (5).

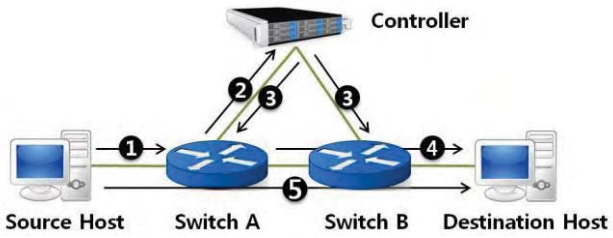

Figure 2. Basic SDN operation of setting up a network flow

2.2 Previous solutions for scalability issue

Some previous works focused on the scalability issue, but they consider another aspect, that is, they try to reduce the number of flow entries [7] [8]. In [7], by using wildcard they can reduce the number of flow entries. Additional, the flow entries of normal switch are stored in the authority switch that has bigger memory. When a flow comes to ingress switch, it will match a partition rule that redirects the message to an appropriate authority switch. In authority switch, the packet matches an authority rule that forwards the packet to the egress switch and it installs a cache rule to the ingress switch. The subsequent packets come to ingress switch will match the cache rule which has higher priority than partition rule. Hence, the ingress switch can forward packet to egress switch directly. This method can keep the packet in the data plane and reduce the delay caused by intervention with the controller.

In [8], instead of using wildcard to reduce the number flow entries, the authors combining the flow entries which have the same path to one flow entry in the middle switch. When a

packet comes to an ingress switch, the ingress switch adds two tags to the packet. A flow tag specifies the flow of packets. And a path tags that is used to specify the path of packets. When the ingress switch forwards a packet to a middle switch, the packet matches a flow entry by the path tag. Therefore, the packet can be forwarded to the egress switch which removes both tags. Instead of storing multiple flow entries that do the same action-forwarding packet to the same path, the middle switch needs to store only flow entries which match the tag and does the same action.

Our proposed solution considers another aspect. We try to expand a flow table logically. However, a flow table of a device has some limitation about increasing the volume as we discuss before. Therefore, instead of increasing the flow table size by hardware, we increase the size of flow table by using software that running on a machine which has much bigger memory.

3. Flow Table Management

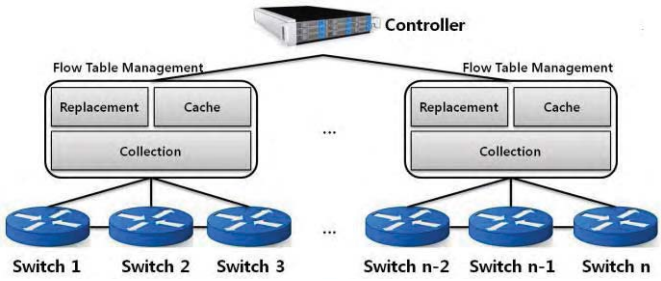

Figure 3. Architecture of Flow Table Management

In this section, we describe the detail design of Flow Table Management. Flow Table Management contains 3 modules; *collection module*, *replacement module* and *cache module*. Collection module is to collect the statistic information of flows. Replacement module is to replace flow entries when controller fails to install a new flow entry because a flow table is full. It answers two questions whether new flow entry should be installed and which old flow entry should be replaced. Cache module stores the flow entries which are not installed to a device by Replacement module. Hence, it can quickly handle the packet which matches with cached flow entries and replaced flow entry. These three modules can be integrated into a controller. However, a controller is a bottle neck of SDN and putting these modules inside the controller may make it worse. According to this, it is preferred to locate a middleware with Flow Table Management between a controller and devices. Figure 3 describes the architecture of Flow Table Management (FTM). FTM can be deployed into multiple middleware nodes and each node operates independently. When the number of devices increases, the number of nodes with FTM can increase to distribute the load of FTM. So that, each FTM node is not overloaded and it prevent from being a bottle neck of SDN. Every transfer message between a controller and a device must goes through FTM. FTM will only process related packet type such as install flow entry and statistic response. With other packet types, FTM simply forwards it to a controller or a device.

3.1 Replacement

Replacement module decides whether a new flow entry should be installed or not and which old flow entry should be replaced. To improve the performance of SDN, we will try to keep high load network traffic in the data plane and let cache module handle low load network traffic. That is, we will try to minimize the number of packets as well as the bandwidth of flows which are handled by cache module. Achieving both goals together is almost impossible. Therefore, we run a simple ranking algorithm to compare the utility between flow entries. When a controller application fails to install a flow entry to the switch, replacement module compares the rank of the new flow entry and installed flow entries in the switching device. If the new flow entry has higher rank than some of the flow entries in the device, replacement module replaces one of lower ranked flow entries with the new flow entry. Equation (1) shows a basic method to calculate a rank of a flow. In this equation, *V* means the load of a flow and *W* means the weight value for a category of a flow.

$$\mathrm{R_f} = \sum_{i=1}^{k} W_i \times S_{if} \quad (1)$$

$$S_{if} = \frac{V_{if}}{\max(V_f)} \quad (2)$$

When replacing flow entries, the *timeout* field of the flow entry can be modified. If the replaced flow entry is related to a high load flow which has a characteristic of long session time, replacement module increases the timeout value of the flow entry. On the other hand, if the replaced flow entry is for a low load flow with a short session time, the timeout value of the flow entry decreases. According to this, replacement module may reduce the number of replacement operations.

3.2 Collection

Collection module periodically sends request to switching devices to collect flow information inside the devices. Then, this information is stored in the local storage of FTM middleware, and it is used in replacement and cache module. In order to save bandwidth and to improve the performance of SDN, collection module tries to use statistic information of communication packets between devices and a controller.

3.2 Cache

Cache module is to handle requests from switching devices. Cache module stores the new flow entries which are not installed into switching devices as well as the replaced flow entries from switching devices. When a device sends a request to the controller, cache module tries to find a matched flow entry. If it exists, cache module sends a response to the device with an instruction to handle the

packet. Otherwise, the request is forwarded to the controller like normal requests.

4. Conclusion and Future Works

In this paper, we propose a solution to solve the flow table size issue in data plane. In order to improve performance of SDN, we propose a flow replacement management middleware with three different modules; cache, replacement, and collection. Especially, to replace flow entries for better performance, replacement module uses a ranking method to sort out the utility of flows.

Currently, we are working on implementing this ranking-based flow replacement method into an SDN application over OpenDaylight SDN controller.

Acknowledgement

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2014R1A1A1007734).

References

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, “Onix: A distributed control platform for largescale production networks,” in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI’10*. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[2] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking.” *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[3] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalag, P. Sharma, and S. Banerjee, “Devoflow: Scaling flow management for high-performance networks,” in *Proceedings of ACM SIGCOMM*, 2011.

[4] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, “A security enforcement kernel for openflow networks,” in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN ’12*. New York, NY, USA: ACM, 2012, pp. 121–126.

[5] “Sdn architecture,” june 2014, accessed: 2014-09-12. [Online].Available: https://www.opennetworking.org/images/stories/downloads/sdnresources/technical-reports/TR SDN ARCH 1.0 06062014.pdf

[6] “Openflow switch specification,” oct 2013, accessed: 2014-09-12. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-specv1.4.0.pdf

[7] Milan Yu, Jenifer Rexford, Michale J. Free Man, and Jia Wang, “Scalable Flow-Based Networking with DIFANE.” in *Proceedings of ACM SIGCOMM*, 2010.

[8] Subhasis Banerjee, and Kalapriya Kannan, “Tag-In-Tag: Efficient Flow Table Management in SDN Switches.” in *Proceedings of 10th International Conference on Network and Service Management* , pp. 109-117, 2014.