A Flow Entry Management Scheme for Reducing Controller Overhead

Eun-Do Kim*, Seung-Ik Lee**, Yunchul Choi**, Myung-Ki Shin**, Hyoung-Jun Kim**

* Broadband Network Technology, UST (University of Science and Technology), Korea

** Protocol Engineering Center, ETRI, Korea

{maniada, seungiklee, cyc79, mkshin, khj}@etri.re.kr

***Abstract*— In this paper, we advocate addressing the communication overhead problem between OpenFlow controllers and OpenFlow switches due to *table-miss* in a flow table. It may cause the communication overhead between controllers and switches because a switch has to send *packet-in* message to a controller for processing table-missed flows. We propose a simple flow entry management scheme for reducing the controller overhead by increasing the flow entry matching ratio. By using an LRU caching algorithm, a switch can keep the flow entries in a flow table as many as possible, and then the flow entry matching ratio can be increased.**

***Keywords*— SDN, OpenFlow, Flow-based networking, Flow entry, Performance**

I. INTRODUCTION

Software-defined networking (SDN) provides network operators to control and manage their own network in a simple and programmable manner. OpenFlow is a communication protocol that allows separation of control- and data-plane of switch functionalities in SDN. Using this protocol, controllers can manage the flow entries in a switch.

In OpenFlow networks, an incoming flow is forwarded according to the action specified in the flow entry in a flow table whose rule is matched with that flow. If an incoming flow does not match one of the rules in the existing flow entries called *table-miss*, the switch informs the controller about the flow properties by sending *packet-in* message to make the controller perform further actions (e.g., inserting a new flow entry relevant to the incoming flow, dropping the flow, etc.).

However, this *packet-in* procedure to deal with *table-miss* may cause the communication overhead between controllers and switches (i.e., controller overhead). So, it may need extra processing time for forwarding packets. As a result, a buffer of a switch may be full if the incoming flows are stayed in a buffer for a long time.

In this paper, we advocate addressing the communication overhead problem between OpenFlow controllers and OpenFlow switches due to *table-miss* in a flow table. We propose a flow entry management scheme for reducing the controller overhead by increasing the flow entry matching ratio with an enhanced but simple buffer management scheme for a switch. Two key ideas of the proposed scheme are as follows:

- A switch keeps the inactive flow entries in a flow table temporarily rather than deleting them.
- A switch deletes the inactive flow entries in accordance with the least-recently-used (LRU) caching algorithm.

With these two ideas, a switch can keep the flow entries in a switch as many as possible, and the controller overhead can be reduced by keeping the flow entry matching ratio high.

In Section 2, we define the controller overhead problem in detail, and introduce the previous works. And then, we propose a modified flow entry management scheme in Section 3. In this Section, the flow table overflow problem which is mentioned in OpenFlow Switch Specification 1.4.0 [3] will be addressed, and the solution applying an LRU caching algorithm is proposed. In Section 4, we describe the tentative simulation results of the proposed scheme which can reduce the controller overhead. Then, the conclusions and the future works are represented in Section 5.

II. PROBLEM DEFINITION

Table 1 shows the main components of a flow entry in a flow table. As depicted, a flow entry contains a set of packet fields to match (*Match Fields*) and the corresponding action for the matched packets (*Instructions*). And it also has *Counters*, *Timeouts*, etc.

A flow entry can be inserted, modified or deleted by a controller, while a flow entry is automatically deleted if there are no matching packets for *idle_timeout* seconds (i.e., inactive flow). Here we tackle two limitations in the current flow entry management scheme for inactive flows:

- A switch may delete the flow entries even if they are matched enough before but not matched recently.
- A switch may delete the flow entries even if a flow table is not fully occupied.

TABLE 1. MAIN COMPONENTS OF A FLOW ENTRY IN A FLOW TABLE

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

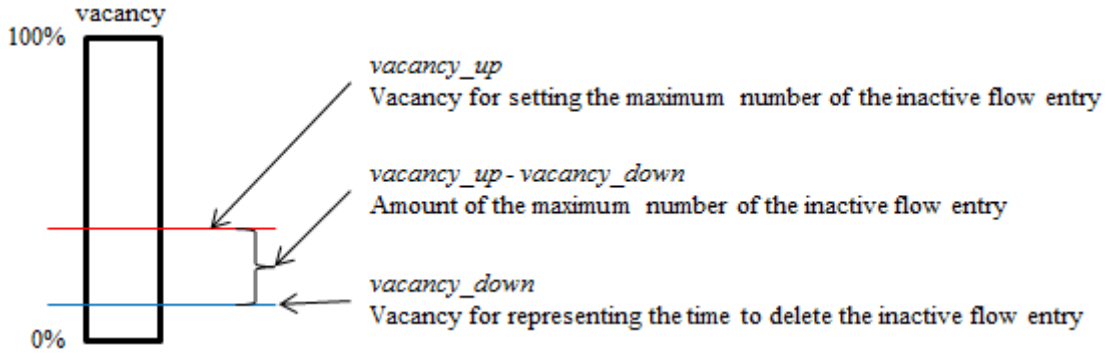

Figure 1. Concept of the vacancy that we propose for applying an LRU caching algorithm

These limitations may result in a low ratio of flow entry matching (i.e., high rate of *table-miss*), and it may cause the communication overhead between controllers and switches. To solve this communication overhead problem, following approaches have been studied:

- DevoFlow [5] uses a selective *packet-in* method by classifying the incoming flows according to their patterns (e.g., *significant flow*, *micro flow*).
- DIFANE [6] uses *authority switches* (i.e., intermediate switches) between controllers and switches which can keep the flow entries temporarily.

While these approaches provide good alternatives to reduce the communication overhead due to many *table-misses*, they still require a sophisticated packet classification algorithm or large changes in the OpenFlow network model. So, they may cause other side effects.

III. Proposed Scheme

We propose a modified flow entry management scheme that increases the flow entry matching ratio. The goal of the proposed scheme is to reduce the communication overhead between controllers and switches by keeping the flow entries in each switch as many as possible. In order to maximize the number of the flow entry in a switch, we keep the flow entry in a flow table as an inactive one even if a packet does not match the flow entry while *idle_timeout* expires. We call this undeleted flow entry inactive flow entry.

Whenever an inactive flow entry is matched by any packet later, it becomes an active flow entry again by resetting the *idle_timeout* and *counter* to their default values. In this case, the controller overhead can be reduced because a switch can insert flow entry without any communications between controllers and switches.

A. Flow Table Overflow

In OpenFlow Switch Specification 1.4.0, the flow table overflow problem is mentioned as follows:

- Most flow tables have finite capacity.
- In previous versions of the specification, when a flow table is full, new flow entries are not inserted in the flow table and an error is returned to the controller. However, reaching that point is pretty problematic, as the controller need time to operate on the flow table and this may cause a disruption of service.
- Vacancy events add a mechanism enabling the controller to get an early warning based on a capacity threshold chosen by the controller (EXT-192). This allows the controller to react in advance and avoid getting the table full.

In OpenFlow networks, a switch sends an error message to a controller when a flow entry has to be inserted, but cannot because all the flow tables are full. If a controller wishes the packet to be sent regardless of inserting a flow entry, then it can also use a *packet-out* massage alternatively. However, it cannot be a fundamental solution because of some problems as follows:

- A new flow entry, which is very important, may not be inserted due to flow table full.
- A controller does not know the flow table is full until it receives an error message from a switch when a flow entry cannot be inserted. So, a controller may need extra processing time.
- After a flow table is full, a controller may receive more and more error messages from a switch since new flows income.

To avoid getting the flow table full in advance, we propose a modified flow entry management scheme applying an LRU caching algorithm.

B. Applying an LRU Caching Algorithm

To avoid the flow table overflow, a switch has to delete the least important inactive flow entry. For this purpose, we choose an LRU caching algorithm which shows the best performance for increasing the flow entry matching ratio. The performance comparisons between some caching algorithms are described in [2].

For applying an LRU caching algorithm with simple modifications on the switch, we use a concept of the vacancy as shown in Figure 1. A concept of the vacancy is newly added in OpenFlow Switch Specification 1.4.0 as follows:

- *vacancy* is current vacancy (%) of the flow table.
- *vacancy_down* is the vacancy threshold when space decreases (%).

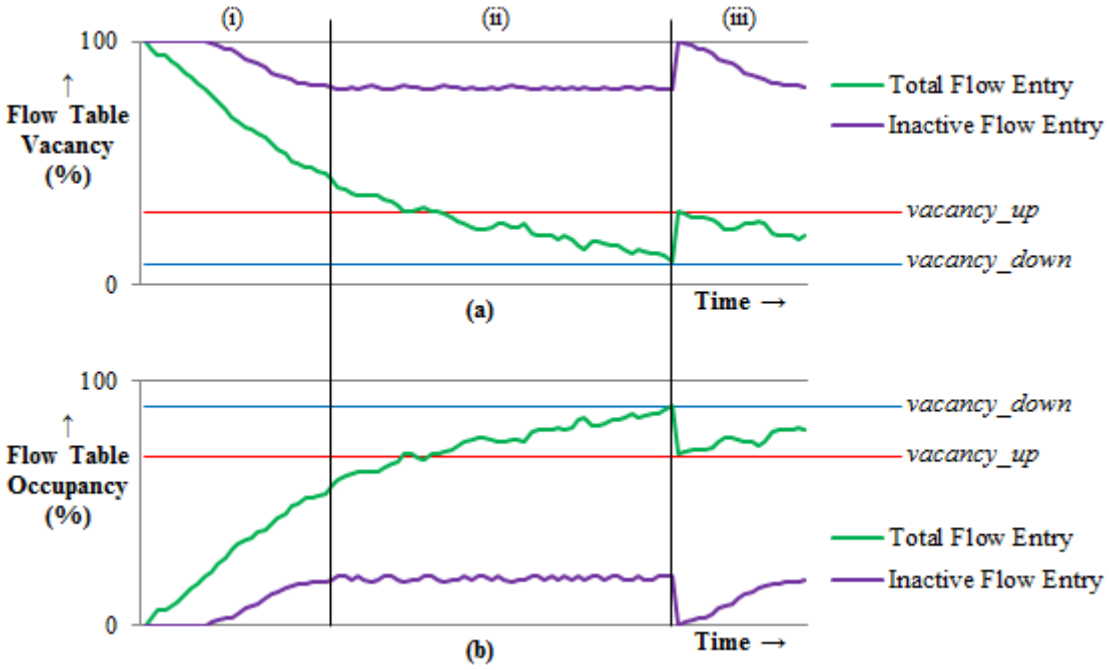

Figure 2. Illustration of the (a) vacancy and the (b) occupancy of a flow table in a switch along the time with a cold start

- *vacancy_up* is the vacancy threshold when space increases (%).
- The fields *vacancy_down* and *vacancy_up* are the threshold for generating vacancy events that should be configured on this flow table, expressed as a percent.

By using this concept of the vacancy, we can apply an LRU caching algorithm for deleting the least important inactive flow entry in a simple manner with a minor modification in OpenFlow switches.

To apply this, inactive flow entries are maintained with their ages by exploiting *counter* field in each flow entry which is used to measure the number of flow matching for the purpose of statistics. When a flow entry goes inactive, a switch resets its *counter* to zero rather than deleting that flow entry, and then a switch increments *counter* of the other inactive flow entries. As a result, the least-recently-used inactive flow entry may have the highest *counter* (i.e., highest *counter* of the inactive flow entries means the number of the inactive flow entry).

In the proposed scheme, the maximum number of the inactive flow entry is set to the amount of *vacancy_up* minus *vacancy_down*. Because the number of the inactive flow entry increases, inserting different flow entries to the switch about the previous flow may be problematic. To delete an inactive flow entry based on LRU caching algorithm, a switch can exploit *counter* field of each flow entry which indicates its age. A switch deletes the inactive flow entry whose *counter* field meets the amount of *vacancy_up* minus *vacancy_down*. By doing this, a switch can keep the number of the inactive flow entry as we set.

If *vacancy* becomes less than *vacancy_down*, however, a switch has to delete some flow entries regardless of their *counter* for avoiding the flow table overflow. In this case, the target flow entries to delete are all of the inactive flow entries. By doing this, a switch can increase *vacancy* again through a simple manner.

IV. Simulation Results

The controller overhead is to be reduced by applying the proposed scheme. Figure 2 shows an illustration of both the vacancy and the occupancy of a flow table in a switch along the time with a cold start. These graphs are described based on the tentative simulation with an LRU caching algorithm by using Open vSwitch [7] and Mininet [8] which are open-source tools for OpenFlow simulations. We implemented the modified behavior of an OpenFlow switch by modifying the source code of Open vSwitch roughly. And then, we construct a topology and generate flows by using Mininet.

For a simple explanation, we represent as a view of the occupancy as well as the vacancy of a flow table in a switch. This process is classified as three stages as follows:

- In stage (i), flow entries are inserted in a switch with a cold start. When a switch starts, default flow entries which are calculated by applications (e.g., firewall, QoS, etc.) are inserted from a controller. And then, the flow entries which set the routing path are also inserted as flows income. As time goes by, the flow entries whose *idle_timeout* expires go inactive, then the number of the inactive flow entry increases. As a result, the flow table occupancy increases (i.e., flow table vacancy decreases) along the time.
- In stage (ii), the number of the inactive flow entry is fixed when it reaches the amount of *vacancy_up* minus *vacancy_down*. It is because a switch deletes the

inactive flow entry whose *counter* field meets the amount of *vacancy_up* minus *vacancy_down* (i.e., least-recently-used inactive flow entry). However, the number of total flow entry changes continuously because new flows may income frequently. A switch is usually maintained in this stage.

- In stage (iii), if the number of the flow entry becomes more than the amount of *vacancy_down* (i.e., *vacancy* becomes less than *vacancy_down*), a switch has to prevent the flow table overflow. In this case, a switch deletes all of the inactive flow entries. The reason to delete all of them at once is for simplifying a switch in accordance with one of the philosophies of OpenFlow networks. After deleting them, a switch is returned to the stage (i) but warm start because it already has enough flow entries to forward packets. By doing this, a switch can avoid the flow table overflow in a simple manner.

V. CONCLUSIONS

In OpenFlow networks, there is the communication overhead problem between OpenFlow controllers and OpenFlow switches due to *table-miss* in a flow table. And the flow table overflow problem is also mentioned in OpenFlow Switch Specification 1.4.0.

In order to reduce the communication overhead from many *packet-in* messages, we proposed a modified flow entry management scheme for increasing the flow entry matching ratio. For this purpose, a switch temporarily keeps inactive flow entries rather than deleting them.

If a flow table is fully occupied with flow entries, a switch deletes the inactive flow entry in accordance with an LRU caching algorithm using a concept of the vacancy of a flow table in a switch which is newly added in OpenFlow Switch Specification 1.4.0. By applying the proposed scheme for increasing the flow entry matching ratio, we can reduce the communication overhead from many *packet-in* messages.

As our future work, we will consider additional issues as follows:

- It is required to evaluate the proposed scheme with realistic packet flows. For this purpose, we will simulate our modified flow entry management scheme by using Open vSwitch and Mininet more detailed. Consequently, we will show the enhanced performance of our proposed scheme as comparing with current scheme.
- If a flow table is fully occupied with active flow entries, a switch has to prevent the flow table overflow in other ways. For this purpose, we will develop a careful replacement algorithm to provide a high flow entry matching ratio by using an LRU caching algorithm with a concept of the vacancy as a threshold.

ACKNOWLEDGMENT

This research was funded by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013.

REFERENCES

[1] Eun-Do Kim, Seung-Ik Lee, Yunchul Choi, Myung-Ki Shin, and Hyoung-Jun Kim, "An Enhanced Flow Entry Management Scheme for OpenFlow," in *Proc. CFI'13*, 2013.

[2] Adam Zarek, "OpenFlow Timeouts Demystified," Univ. of Toronto, Toronto, Ontario, Canada, 2012.

[3] *OpenFlow Switch Specification Version 1.4.0*, ONF (Open Networking Foundation), Available: https://www.opennetworking.org/, October. 2013.

[4] *Software-Defined Networking: The New Norm for Networks*, ONF (Open Networking Foundation), Available: https://www.opennetworking.org/, April. 2012.

[5] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, and Praveen Yalagandula, "DevoFlow: Scaling Flow Management for High-Performance Networks," in *Proc. SIGCOMM'11*, 2011.

[6] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang, "Scalable Flow-Based Networking with DIFANE," in *Proc. SIGCOMM'10*, 2010.

[7] Open vSwitch: An Open Virtual Switch. [Online]. Available: http://openvswitch.org/

[8] Mininet: An Instant Virtual Network on your Laptop (or other PC). [Online]. Available: http://mininet.org/

Eun-Do Kim was born in Seoul, Korea, in 1987. He received the B.S. degree in Applied Physics from the Hanyang University, Ansan, Korea, in 2012, and he is an integrated M.S. and Ph.D. student in Broadband Network Technology of the UST (University of Science and Technology), Daejeon, Korea, since 2012.
In 2012, he joined the Protocol Engineering Center, ETRI, Korea, as an UST graduate student and his current research interests include SDN and OpenFlow protocol.