A Zero Flow Entry Expiration Timeout P4 Switch

Cheng-Hung He[1], Brian Y. Chang[1], Suchandra Chakraborty[2], Chien Chen[2], Li Chun Wang[1]
[1]Department of Electrical and Computer Engineering, [2]Department of Computer Science
yukinari.eed03@g2.nctu.edu.tw; brian0296.eed03@g2.nctu.edu.tw; suchandratutul.cs01g@nctu.edu.tw;
chienchen@cs.nctu.edu.tw; lichun@g2.nctu.edu.tw
National Chiao Tung University, Hsinchu, Taiwan

ABSTRACT

Existing OpenFlow based flow expiry mechanisms rely on a fixed timeout after which the switch proactively removes the flow entries from its flow table. Assigning an appropriate value to this timeout presents a tradeoff between efficient flow table utilization and controller overhead. We propose a simple addition of a zero flow entry expiration timeout scheme to idle timeout mechanism on a P4 based switch that identifies the last packet of a TCP flow, and removes the corresponding flow entry from the flow table. It enables connections to be deleted from the flow table right after it finishes, instead of waiting to expire after it reaches the timeout threshold and causing unnecessary flow table occupation.

CCS CONCEPTS

• **Networks → Programmable Networks**;

KEYWORDS

SDN, TCP, P4, Idle Timeout, TCAM

1 INTRODUCTION

In order to build large scalable SDN networks, the OpenFlow [4] switches have to support a large number of concurrent flows, which depend on the number of flow entries in their flow tables. However, the flow table is limited due to the constraints of Ternary Content Addressable Memory (TCAM) [1]. The controller manages the switch flow table by adding and removing flow entries based on two timeout values: hard timeout and idle timeout. It presents a tradeoff between efficient flow table utilization and controller overhead. To reduce unnecessary TCAM occupation, the switch removes the corresponding flow entry after the timeout expires without matching traffic.

However, flow entries sometimes still unnecessarily occupy the flow table even after the end of the original flow. In addition, assigning a smaller time out reduces the unnecessary occupation of the flow table, but introduces further controller overhead as the controller may need to install the same flow entry multiple times based on the packet-in events generated by the switch [3].

In this paper, we propose the idea of detecting the FIN and RST packet of a TCP flow and remove the corresponding entry from the flow table when detected. Our work aims to achieve optimal flow table utilization without extra workload for a controller.We implement a simple zero flow entry expiration timeout scheme that leverages the potential of a P4 based switch to detect the FIN and RST packet of a TCP flow and remove the corresponding flow entry from the flow table, eliminating unnecessary occupation of the flow table to achieve almost zero timeout without overloading the controller.

In the TCP Finite State Machine, TCP sessions can be closed by sending combinations of FIN and RST packets. In our work, connections are removed from the flow table through the detection of FIN and RST packets, avoiding taking up flow table capacity after the connection is finished. By adding the detection of FIN and RST packets to idle timeout, our scheme achieves an improvement in flow table occupation compared to pure idle timeout. When there is a high proportion of short connections, such as in data center networks, our method, combined with idle timeout, will prevent short connections from continuing to occupy the flow table after the connection is closed.

As for UDP and half-closed TCP connections, by the means of including idle timeout, we can still assure that the connections will eventually reach the timeout threshold and be removed from the flow table.

We evaluate the performance of the proposed mechanism using Barefoots Tofino switch and monitor its performance in the number of packet-in events as well as flow table occupation. The remainder of this paper is organized as follows. Section 2. describes the architectural design, Section 3. presents the experimental results, and Section 4. concludes the paper.

SOSR'18, March 2018, Los Angeles,CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5664-0/18/03…$15.00
https://doi.org/10.1145/3185467.3190785

2 ARCHITECTURE DESIGN

We present the overview of our architecture of the proposed in this section and describe the P4 program's mechanism.

The P4 switch has two tables: Forward and Learn. The Forward table is first applied for proper routing and matching any static forwarding entries. Then the Learn table detects whether the ongoing packet is a FIN or RST packet right after the P4 program parses the TCP header. If so, a message is sent to the controller (Open Networking Linux) to give a message to delete the entry of that flow from the flow table.

3 EXPERIMENTAL RESULTS

We now evaluate our proposed design using the Barefoot Tofino hardware switch with two experiments [2] to compare our scheme with the flow table usage and packet-in events of hard and idle timeout. The traffic is generated by Iperf with the flow duration chosen through a Normal distribution, and the arrival time between each flow determined through a Poisson process. The arrival rate of TCP flow is =3(1/s), mean flow duration is 2.5s, and number of TCP flows is 600 during the evaluation period. Evaluation tests are performed on the Barefoot Tofino switch and monitor for 150 seconds.

Figure 1 shows the change of flow table entries over time. The timeout values of all schemes are set to 2 seconds. As shown in Figure 1, compared to idle timeout and hard timeout, our scheme achieves much lower TCAM occupation, as the curve of our scheme is always below the curve of the hard timeout and idle timeout schemes.

In Figure 2, different timeout values ranging from 1s to 4s are used to depict their packet-in rate and peak TCAM occupation. We can discover that our scheme achieves fewer average packetin events than hard timeout, and has obvious improvement in TCAM occupation compared to both timeout schemes. Our scheme has improvement in peak TCAM occupation compared to idle timeout, because besides exceeding the idle timeout threshold, it only removes entries upon detecting FIN/RST packets, which ensures that a connection is already terminated. Therefore, if entries are not dropped by the idle timeout mechanism, there will only be exactly one extra packet-in event due to the FIN/RST packet, but on the other hand greatly reduces TCAM occupation since the entries will be removed immediately after a connection is finished. Overall, compared to the other schemes, our work achieves a better tradeoff as having the advantage of idle timeout schemes better packet-in rate and a much better TCAM occupation status compared with both idle and hard timeout.

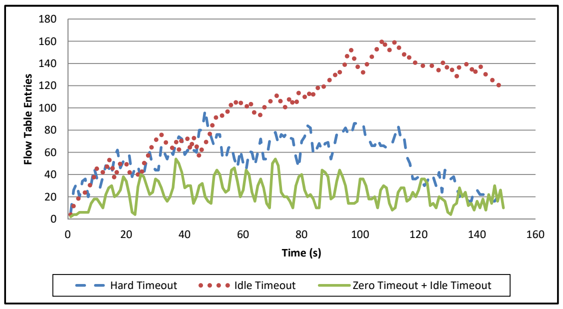

Figure 1: Comparison of TCAM occupation with time.

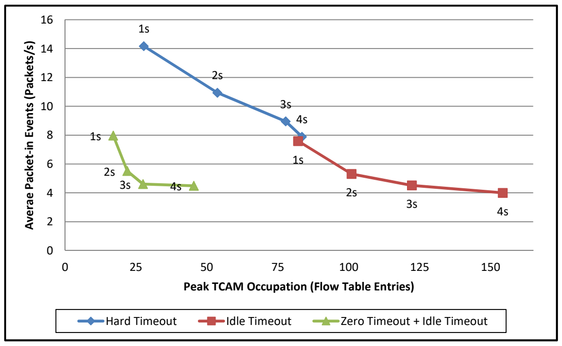

Figure 2: Comparison of TCAM occupation with pkt-in rate.

4 CONCLUSION AND FUTURE WORKS

We proposed a design of a flow entry removal technique for TCP flows on a simple P4 switch that detects the last packet of a TCP flow and deletes the corresponding flow entry to reduce unnecessary TCAM occupation. Experiments were performed to evaluate its performances. Future works will be to test and evaluate this scheme with larger scale realistic workloads.

ACKNOWLEDGMENTS

This research is supported partially by the Chunghwa Telecom Laboratories and the Ministry of Science and Technology of Taiwan under Grant: MOST 106-2221-E-009-035.

REFERENCE

[1] V. Mann S. Bhattacharya A. Vishnoi, R. Poddar. 2014. "Effective switch memory management in openflow networks". In *in ACM DEBS 2014.* pp.177–188.

[2] X. Luo Y. Jin H. Zhu, H. Fan. 2015. "Intelligent timeout master: Dynamic timeout for sdn-based data centers". In *in IFIP/IEEE International Symposium on Integrated Network Management (IM) 2015.* pp.734–737.

[3] Flowmaster K. Kannan, S. Banerjee. 2014. "Early eviction of dead flow on sdn switches". In *in International Conference on Distributed Computing and Networking (ICDCN) 2014.* pp.484–498.

[4] H. Balakrishnan G. Parulkar L. Peterson J.Rexford S. Shenker N. McKeown, T. Anderson and J. Turner. 2008. "Openflow: enabling innovation in campus networks". In *SIGCOMM Comput. Commun. Rev., 38(2).* pp.69–74.