Taming the Flow Table Overflow in OpenFlow Switch

Siyi Qiao, Chengchen Hu, Xiaohong Guan, Jianhua Zou
Xi'an Jiaotong University
Xi'an, China
qsy2011815@sina.com, huc@ieee.org, {xhguan, jhzou}@sei.xjtu.edu.cn

CCS Concepts

•**Security and privacy** → **Systems security;** •**Networks** → *Network architectures; Routers;*

Keywords

SDN; Group table; flow table; overflow ; table-miss

1. INTRODUCTION

Software Defined Networking (SDN) is an emerging network architecture, which decouples the control plane from the data plane and operates the global network with elaborate abstraction [1]. The flow table plays an important role in an OpenFlow Switch (OFS) [2] and is the key to support the SDN/OpenFlow abstraction. However, to provide wire-speed processing, fast memory (*e.g.*, TCAM, QDR, SRAM) is utilized to form the flow table. Unfortunately, the development of such kind of fast memories is far behind the hungry requirement on its usage. As a result, the flow table installed in OFS has large risk to be overflowed, possibly leading to large number of packet-in/packet-out messages between OFS and controller.

As shown in Figure 1(a), an incoming packet from a flow is processed according to the action specified in the according flow entry in the flow table(s). If no entry is matched in the flow table, a packet-in message querying how to process the packet will be sent to the controller. If the number of active flows usually exceeds the maximum number of entries in the flow table, the table-miss events are not avoidable. Then, controller will delete an active flow and add this new flow, meanwhile, that active flow becomes to a new flow. The packet-in message revolves in the secure channel uncontrollably. On the contrary, if we ignore the unmatched packet, that is means we stop serving and it is unfair to the later flow.

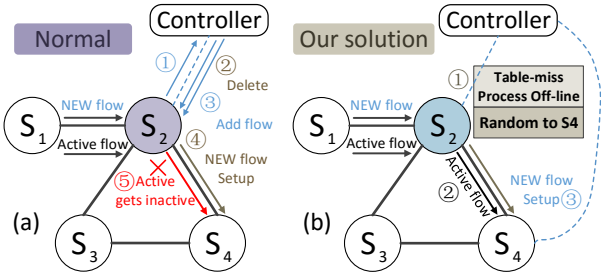

Figure 1: Classical and FTS structure

In this paper, we investigate how to mitigate the overhead when occurring table-miss events based on the phenomenon of uneven flow table distribution. The basic idea is to distribute the packets facing table-miss event in heavily loaded switch to other lightly loaded switches instead of triggering packet-in message always in hot switches. The new mechanism proposed in this paper to handle the table-miss event is named Flow Table Sharing (FTS).

2. DESIGN OF FTS

Example operation. Figure 1(b) shows an example topology with 4 routers and S2's flow table is just used up. To ensure that FTS prevents the storm of control messages caused by the packet-in event, the S2 has to stop sending packet-in messages to SDN controller. The new flow will be handled by the Off-line table-miss process, *e.g.*, forwarding it to S4 randomly that has spare flow entries. After that, S4 can complete the policy routing by normal packet-in process.The conceptual simplicity of FTS idea hides two significant challenges. 1) How to select a right port randomly by SDN switch. 2) How to make this progress "pipeline-able" in a general SDN switch without changing its Hardware.

In the design and implementation of FTS, we address these two challenges by a new external **select** algorithm and **group** actions. The SDN switch can process a packet through the

This paper is supported in part by the National Natural Science Foundation of China (61221063, 61272459, U1301254), 863 High Tech Development Plan (2012AA011003) 111 International Collaboration Program of China, Program for New Century Excellent Talents in University (NCET-13-0450) and the Fundamental Research Funds for the Central Universities.

SIGCOMM '16, August 22-26, 2016, Florianopolis , Brazil

DOI: http://dx.doi.org/10.1145/2934872.2959063

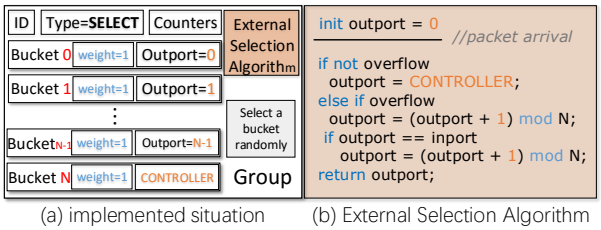

Figure 2: The configuration of group table and the external selection algorithm

specified group that is one of pipeline steps in a standard OpenFlow SDN switch originally. The external user switch-computed select algorithm [2] (*e.g.*, hash on some user-configured tuple or simple round robin) is allowed by the **SELECT** type of group table, according to the Spec. OpenFlow1.3. This select algorithm can help us assign an executable bucket that contains actions for the new flow. To implement FTS, one flow table entry and one group table are required. If the incoming packet can not match any high priority entry, it will be managed by the flow table entry reserved by FTS.

We still retain the table-miss flow entry which action is go-to FTS Group table. As Figure 2(a) shows, the buckets in group table contain N actions to all the corresponding neighbors of this switch and 1 action to CONTROLLER. It will not select the CONTROLLER bucket until the flow table has spare space for new flows. If we set the weight of a bucket by 0, the actions of this bucket will be disabled. It is equal to today's general process of table-miss event. The work-flow of user algorithm is shown in Figure 2(b). The external selection algorithm needs to check the flag of flow table overflow and calculate the value of outport simply and fast.

3. EVALUATION

Performance test:we first build a switch in mininet, and measure the number of control messages generated by setting up a new flow transfer(TCP, UDP) when the flow table of the switch is overflow, as well as the packet loss rate and the average delay. We now demonstrate that the FTS can reduce the storm of control messages and RTT time during the flow table overflow period.

Table 1: Performance comparison between FTS and ordinary switch

	UDP/loss	TCP/loss	RTT(ms)
optimal	37 / 0%	37 / 0%	0.227
our solution	43 / 0%	43 / 0%	7.56
overflow	379 / 15%	2199 / 0%	768

Table 1 summarizes three kinds of experiment results that we compare to the optimal and the worst situations: UDP/loss means the number of control messages / packets loss rate during the time we transit a UDP network flow. RTT(ms) shows the average packets forwarding delay after building the flow in different situation (optimal, our solution, and overflow, where "optimal" is the situation that the flow table has enough free space for following new flow).

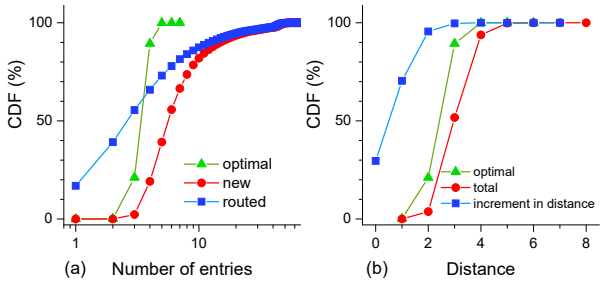

Figure 3: Accumulate distribution function of additional flow table resource usage for set a flow and the hops between src and dst for each packet, when overflow.

Cost test:we evaluate the flow table demand in the optimal way that all switches have enough flow table resources and set this result as the control group. Then, on the one hand, we evaluate the additional flow table demand which is required by rebuilding the interrupted flow, when the FTS try to fix the problem caused by the overflow. On the other hand, we evaluate the total flow table consumption which is required by building the new transmission for the first time, after the overflow happened. As Figure 3(a) shows, blue line indicates the rebuilding situation and the red line indicates the new flow. We find that FTS consumes more flow table resources and its average usage is 8.7 while the optimal is 5.9.

The green line in Figure 3(b) indicates the minimum (optimal) distance between any two points. While the red line indicates the distance passed by the packet which is handled by FTS after overflow. The blue line indicates the extra distance caused by FTS. We find that it will ensure 95 percent probability that the extra distance is less than 2.

4. CONCLUSION

This paper presents a new architecture named FTS to overcome the big performance disaster caused by the flow table overflow in SDN switches. It has several advantages: 1)FTS reduces both control message quantity and RTT time by one orders of magnitude compared to current state-of-the-art OpenFlow table-miss handler. 2)We show the validity and fastness of the external user switch-computed select algorithm. And 3)It is Easy to implement, easy to control and the current state-of-the-art OpenFlow table-miss handler is a special case of FTS.

5. REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[2] O. S. Specifiation. v1. 5, open network foundation, september 27, 2013.