Development of system for mapping of an arbitrary number of OpenFlow tables to single OpenFlow table switches in software-defined networks

Vladislav Korolev, Moscow State University, Moscow, Russia, vkorolev@lvk.cs.msu.su

Alexander Shalimov, Moscow State University, Moscow, Russia, ashalimov@lvk.cs.msu.su

Introduction

Software-defined networking is an approach where the management level is separated from the data transmission level. Now there is logically one centralized controller, which takes control over other devices. One way of the SDN technology realization is OpenFlow. One of basic factors of OpenFlow structure is existence of Flow-tables where the rules applicable to the processed packets are written down. As more flow tables are on switches, as more functional a network managed program can be. At the moment full-fledged hardware OpenFlow switches have not yet appeared. Currenly we mostly have OpenFlow-enabled classical switches with limited number of flow tables. Thus, it becomes valuable to map arbitrary number of flow tables to the switches with limited number of them.

Problem

Work purpose

The purpose of this work is development of the system of translating any set of OpenFlow tables into single table in a general network device. The system has to support the rules containing three main fields: the field for comparison, the field of actions, and a priority. Besides, the system should support addition and removing of rules.

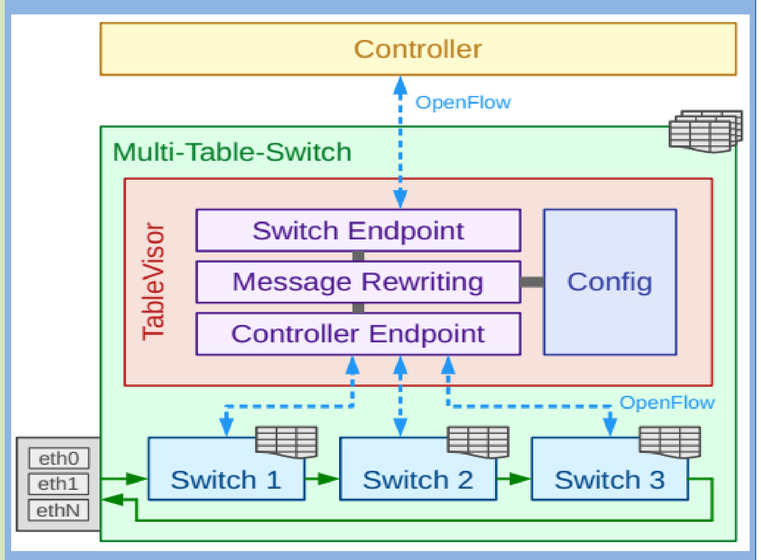
Problem definition

Problem definition is to develop, realize and research an algorithm of translating any set of OpenFlow tables into single table in a general network.

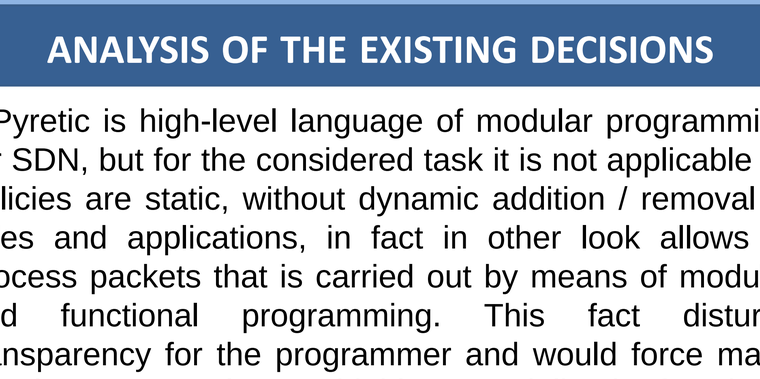
Justification of relevance

1. OpenFlow-enabled classical switches have the small number of OpenFlow tables – one or two usually.
2. Even during developing OpenFlow[4] switches from scratch using network processors developers get problems with implementation of several OpenFlow tables. As more flow tables requiers as more additional passes the packet loops through network processor. This overall reduces the total bandwidth of the device.

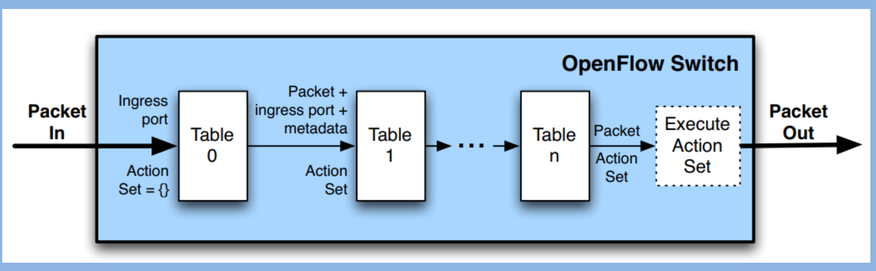
TableVisor Architecture



SDN with Pyretic



OpenFlow Switch



Analysis of the existing decisions

Pyretic is high-level language of modular programming for SDN, but for the considered task it is not applicable as policies are static, without dynamic addition / removal of rules and applications, in fact in other look allows to process packets that is carried out by means of modular and functional programming. This fact disturbs transparency for the programmer and would force many developers to learn highly specialized functional language.

TableVisor represents the proxy server written in the Java [5] language widespread and known to many developers, has an opportunity to emulate the switchboard with big functionality for rather small financial sum in comparison with the physical switchboard having necessary functionality. Also ensures functioning in one system of devices of different producers. Directly does not solve our problem, but uses useful approach of the proxy server.

Approach to the decision

The input parameters of the proposed algorithm: a set of tables is given; we process them from right to left, exactly from the last table to the first one. Every table has a type, which depends on the last rule: drop the packet or go to the next. These two types are D and G respectively. Every time only two tables are considered: created on the last step with the next one (on the first step, the last and pre-last tables are considered). The type of the formed table on each step coincides with type of the current table (left of considered). If the type of the current table is D, then the forming table is the association of considering tables; if the type is G, then the forming one is the association of considering ones plus the formed table, which should be added down the forming table. When we want to have a combination of two tables we look at couples of rules: fix top rule of the left table and compare to all rules from the right one from top to down, then fix the second rule of the left table – compare with all from right and so on. As a result, we will compare every rule with all another rules; the order of a pass is set. Make intersection the fields for comparison (match fields) according to the name: if intersection is empty or there is a crossing, but with equal values, the rule with association of fields for comparison is added; if there are various values of the respective fields in the crossing then the new rule should not be formed.

At addition of the new rule the field of actions (action) will be formed so: add the actions union of considering rules taking into account that at nonempty intersection there will be fields with value from the second rule in association. In the final table priorities are placed from below - up increase.

Experimental evaluation

For carrying out experiments and a research of the developed algorithm the program on C++ was written. Input parameters are various txt files containing tables of different configurations with different sizes. These tables are created manually. The configuration is a set of fields for comparison and actions: actions can be intersected by different rules and can be absolutely different. It means arbitrary actions are considered. Besides, various order of types of tables, for example, of DDGGDD, GGDDDGG, GDGDGD, DGDG, GGGGDD, DDDG, GDGGDDG was considered. The correctness of work of the developed algorithm is confirmed theoretically as there will be the rules containing all possible scenarios of packet processing in the final table. The final rule priority corresponds to reality by the construction of final table. Besides, the correctness of results was manually confirmed on the considered 20 tests. In all tests the final table was constructed correctly. It is possible to notice that the number of rules of the final table many times higher from the number of rules in an original set of tables.

On the basis of a research it is possible to conclude that at increase in total of rules from this set of tables, there is an exponential explosion, otherwise, more the more values the algorithm should touch, the number of rules in the final table increases quicker. There are several ways to optimize algorithm. For example, it is possible to remove of the repeating rules. It is worth noticing that the number of rules in the final table depends not only on the total number of rules at the beginning, but also on the structure of these rules. There are some situations when two rules have many intersections in the name of fields. In that case the probability of not adding the new rule to the final table is greater.

Conclusion

The developed algorithm of display of a set OpenFlow tables in one can expand functionality of TableVisor if it is introduced between OpenFlow the controller and TableVisor the proxy server. The algorithm was developed with acceptance of some restrictions: consideration only write_action of actions.

The following steps on improvement of an algorithm are support of apply_action, conducting testing on real data, integration into TableVisor and an optimization of the rule workaround for better performance of devices.

Table 1. Experiments

Tables quantity	Overall rule quantity	Rule quantity after algorithm
3	20 (6, 6, 8)	307
5	20 (4, 4, 4, 4)	373
4	30 (5, 5, 6, 4)	652
3	40 (17, 13, 10)	2038
4	50 (4, 16, 15, 15)	7481
4	50 (15, 15, 16, 4)	7293
4	60 (20, 3, 4, 33)	7700

References

1. Gebert S. et al. Table visor: An emulation layer for multi-table open flow switches //Software Defined Networks (EWSDN), 2015 Fourth European Workshop on. – IEEE, 2015. – C. 117-118.
2. Geissler S. et al. Tablevisor 2.0: Towards full-featured, scalable and hardware-independent multi table processing //Network Softwarization (NetSoft), 2017 IEEE Conference on. – IEEE, 2017. – C. 1-8.
3. Reich J. et al. Modular sdn programming with pyretic //Technical Reprot of USENIX. – 2013.
4. Nygren A. et al. OpenFlow Switch Specification, Version 1.3. 4 (Protocol version 0× 04), Mar. 27, 2014 //Open Networking Foundation.(Part 1 of 2). – C. 1-84.
5. The continued TableVisor project – JTableVisor: https://github.com/lsinfo3/JTableVisor.

«Modern Network Technologies, MoNeTec- 2018»