2017 ACM/IEEE Symposium on Architectures for Networking and Communications

Rule Compilation in Multi-Tenant Networks

Khalil Blaiech
UQAM, Montreal
khalil.blaiech@gmail.com

Salaheddine Hamadi
UQAM, Montreal
hamadi.salaheddine@gmail.com

Stefan Hommes
University of Luxembourg
stefan.hommes@uni.lu

Petko Valtchev
University of Luxembourg
petko.valtchev@uni.lu

Omar Cherkaoui
UQAM, Montreal
cherkaoui.omar@uqam.ca

Radu State
University of Luxembourg
radu.state@uni.lu

CCS Concepts

•**Networks** → **Data path algorithms; Programmable networks;**

Keywords

OpenFlow, multi-tenant, rule compression

1. INTRODUCTION

Packet forwarding in Software-Defined Networks (SDN) using OpenFlow [5] relies on a centralized network controller to enforce network policies expressed as forwarding rules. Rules are deployed as sets of entries into the tables of network devices. Deploying them onto heterogeneous set of devices is strongly bounded by the respective table constraints (size, lookup time, etc.) and forwarding pipelines. Hence, it is important to minimize the overall number of entries in order to both reduce resource consumption and speed up the search. In this work, we present a compression mechanism for rules of diverging origins that minimizes the number of entries. Since it exploits the semantics of rules and entries, our compiler fits a heterogeneous landscape of network devices. We evaluated implementations of our compiler for both software and hardware switches on a realistic test bed.

Similar work in this area is proposed by *TCAM Razor* [4], that minimises TCAM entries by using compression. *Bit Weaving* [6] groups similar TCAM entries based on its action in order to identify and merge rules together. The *Palette* [2] distribution framework is decomposing large SDN tables into small ones by exploiting shared resources among different connections. The order-independence of classifiers with regards to the covered rules was used as a property in *SaxPac* [3] to build a more efficient classification scheme.

2. MULTI TENANT NETWORKS

Let a *rule* R be a set of match fields $R = \{h_1, ..., h_n\}$ with its corresponding action(s). A *match field* represents a part of a rule that is matched with the respective packet descriptor field (e.g. source IP address (sip)). As control plane entities (e.g. tenants) are able to deploy rules on the network independently, conflicts may arise, e.g. two rules with overlapping match fields but different actions. Another type of conflict are shadowing rules, i.e. the traffic described by one rule gets covered by a rule having a more general description. To avoid such conflicts and ensure overall correctness, we implemented a rule validation mechanism.

In case that a rule defines a general state and can be assimilated to rules of other control entities, the compiler detects a conflictual case and enables rule differentiation which consists of adding additional fields to the rule to make it more specific to its own tenant. We posit that each tenant has a set of exclusive differentiators known by the compiler, which define tenant domains, i.e. a dedicated set of match fields such as IP address range, vlan etc. Thus, a received rule R from tenant T_i should match the differentiators (e.g. MAC/IP range, vlan id) used by the tenant. For example, if a rule R defines a drop action applied to flows that specify an sip=10/8 and the tenant T_i defines a differentiator D={sip=10.0/16}, then the rule is transformed to $\overline{R} = D \cap R$={sip=10.0/16}. Otherwise, if the rule would remain unchanged and another tenant T_j defines a rule with D={sip=10.1/16}, R will affect the flows corresponding to T_j as well, meaning all its flows are dropped.

Rule differentiation may give rise to several adapted versions. In case that a rule a R_i with sip=10/8 belongs to a tenant that defines two differentiators for vlan such as D_1={vlan=10} and D_2={vlan=20}, it is separated into rules $\overline{R}_{i.1}$={sip=10/8, vlan=10} and $\overline{R}_{i.2}$={sip=10/8, vlan=20}. At this step, we use trivial algorithms inspired by the proposal in [7] for rule validation and differentiation.

3. RULE COMPILATION

Our goal is to optimize the management (storage/lookup) in switch tables of OpenFlow rules that might have independent provenance. To address the issue in a disciplined manner, the semantic structure of rules need to be made explicit so that common parts in rule definitions can be recognized and mapped to table entries with no redundancy. Rule compilation is the task of organizing a set of rules into compact lookup structures by exploiting their common match fields.

As an illustration, assume the rules shown in Table 1. With a naïve approach, five tables will be used to store them resulting in ca. 40 entries all of whom will be tested against a packet header (in the worst case). Obviously, some entries would be redundant, e.g. the destination IP address constraint dip=10.1/16 which is shared by rules R_1, R_4, and R_7. To avoid this, one should factor out common rule parts: here, this would further pinpoint protocol=TCP, sip=172.168.1/24, etc. Yet factoring out has a cost due to combinatorics of match fields, hence the need for a disciplined approach. The goal is to efficiently detect groups of rules with the associated sets of common match fields, e.g. R_2 and R_3 which share sip=172.168.1/24 and vlan=6. At

978-1-5090-6386-4/17 $31.00 © 2017 IEEE
DOI 10.1109/ANCS.2017.34

lookup time, such a group must be easily accessible so that a potentially matching packet could be directed at minimal cost. Thus, a "key" is be required which discriminates the group against the rest of the rules. Here, $\{R_3, R_6\}$ is selected from the rest by in_port=1, whereas $\{R_2, R_3\}$ would need two match fields (sip=172.168.1/24 and vlan=6). Using such keys, one can build a lookup structure in the form of an index: a group is the value retrieved by its key. Yet determining the optimal split of the overall rule set into smaller groups and the subsequent key computation are non-trivial problems (in fact, hard problems).

To address the above algorithmic problems, we apply a formal concept analysis (FCA) based approach [1]. FCA is a mathematical framework for qualitative data analysis: It studies the implicit clustering structure behind an object-to-attributes cross-tables. FCA algorithms output a hierarchy of bi-clusters, a.k.a. *concepts*, ordered by set inclusion. Concepts are pairs made of an object set, a.k.a. *extent*, and an attribute set, a.k.a. *intent*, where both sets are strongly correlated. Here, to apply FCA, we substitute rules for objects and match fields for attributes (e.g., ($\{R_2, R_3\}$, {sip=172.168.1/24, vlan=6}) is a concept). Beside concept intents, modern FCA algorithms would also compute minimal generator sets which are, in the general case, subsets of the intent [8]. Under specific conditions, i.e. absence of comparable concepts, they can work as keys that discriminate the encompassing concept among a set of other concepts. Our rule compiling solution amounts to: (1) extracting the concepts of the rules-to-match-fields dataset together with generators, (2) selecting a subset of them that represent a good trade-off between minimal number and minimal overlapping, (3) selecting a generator for each concept, and (4) organize the concepts into a trie with respect to their keys.

4. RESULTS

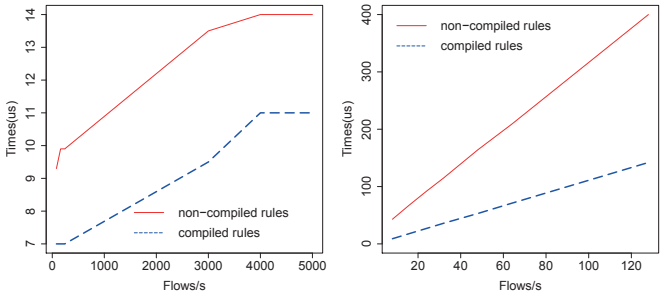
We evaluate the packet forwarding performance by examining the end-to-end delay of packet processing for a software switch and hardware switch (see Figure 1a and 1b). For the software switch, a Kalray MPPA-256 processor chip[1] was used, and a Netronome NFP-3200[2] to handle the compiled rules of hardware switches (e.g. ToR switches). Both represent real high performance switches, and we compare these measurements to a non-compiled control flow program.

Switches based on flow processors achieve a low packet processing latency, but compiled control flow program provide an even lower latency when compared to non-compiled control flow programs. Results show a pay-off up to 50-70 % with a decreasing latency of 50-100 μs for software-based switches, and a pay-off of 29 % and decrease in the latency of 2-3 μs for hardware switches.

[1]http://www.kalrayinc.com/
[2]https://netronome.com/

Table 1: Example rule set. A missing table entry can be considered as a wildcards field in OpenFlow (*).

Rule	sip	dip	vlan	in_port	Prot.
R_1	172.168.1/24	10.1/16	5		TCP
R_2	172.168.1/24	10.2/16	6	2	UDP
R_3	172.168.1/24	10.3/16	6	1	
R_4	172.168.2/24	10.1/16	6	3	
R_5	172.168.2/24	10.2/16	9	4	
R_6	172.168.2/24	30.1/16	7	1	TCP
R_7	172.168/16	10.1/16	5		
R_8	192.18.0/24		9	5	



(a) Hardware switch. (b) Software switch.

Figure 1: Packet forwarding performance for a set of 10,000 rules (non-compiled).

5. CONCLUSION

We designed, implemented and tested a rule compiler for multi-tenant networks that is able to compress forwarding rules to reduce resource consumption of table entries and to speed up the search process for packet classification. Our compiler exploits results from formal concept analysis and network calculus. Its key benefits include ensuring correctness conditions such as isolating tenants from each other and from critical services, optimizing switch forwarding table usage and maintaining high performance packet processing.

6. REFERENCES

[1] B. Ganter and R. Wille. *Formal concept analysis: mathematical foundations*. Springer, 1999.

[2] Y. Kanizo, D. Hay, and I. Keslassy. Palette: Distributing tables in software-defined networks. In *2013 Proc. IEEE INFOCOM*, pages 545–549, 2013.

[3] K. Kogan et al. Sax-pac (scalable and expressive packet classification). In *Proc. of the 2014 ACM Conference on SIGCOMM*, -, pages 15–26, NY, USA, 2014. ACM.

[4] A. X. Liu, C. R. Meiners, and E. Torng. Tcam razor: A systematic approach towards minimizing packet classifiers in tcams. *IEEE/ACM Transactions on Networking*, 18(2):490–500, April 2010.

[5] N. McKeown et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[6] C. R. Meiners et al. Bit weaving: A non-prefix approach to compressing packet classifiers in tcams. *IEEE/ACM Transactions on Networking*, 20(2):488–500, April 2012.

[7] S. Natarajan et al. Efficient conflict detection in flow-based virtualized networks. In *Computing, Networking and Communications, 2012 Intl. Conf. on*, pages 690–696. IEEE, 2012.

[8] L. Szathmary et al. Efficient vertical mining of frequent closures and generators. In *Advances in Intelligent Data Analysis VIII*, pages 393–404. Springer, 2009.