RETCAM: An Efficient TCAM Compression Model for Flow Table of OpenFlow

Chaoqin Zhang, Penghao Sun, Guangwu Hu, and Liang Zhu

Abstract: **OpenFlow is a widely adopted dataplane protocol in software-defined networking (SDN). However, the expansion of supported match fields in OpenFlow brings additional pressure to the storage space of ternary content addressable memory (TCAM) in physical device, since the arbitrary wildcard support in the match field of OpenFlow relies heavily on TCAM for looking-up speed. In this paper, a mathematical model aiming at the storage space reduction of the flow table in TCAM is presented, which is named as RETCAM. RETCAM analyzes the relationships among all the match fields and then categorize the redundancy among different fields into three types. Based on the three redundancy types, three compression algorithms named as inter-field merge, field mapping and intra-field compression are presented. The outcomes of each compression algorithm are flow entries with smaller bit-width which is sent to TCAM for flow matching. In this way, the flexibility of OpenFlow is not harmed, thus maintaining the function integrity of the original flow table. Simulation at the end shows that RETCAM saves almost about 60% of TCAM space for a given flow table with no damage to the function integrity of OpenFlow, and the compression performance stands stable with the increase of flow table size.**

Index Terms: **Compression, OpenFlow, SDN, storage space optimization, TCAM.**

I. INTRODUCTION

RE cently, an innovative network architecture software defined networking (SDN) [1] has been widely adopted in various environments. With the separation of control plane and forwarding plane, SDN provides network users and researchers with much more flexibilities, such as user-defined flow control and open programmability, which propel greatly the revolution towards future networks. To achieve such flexibilities of upper layer, SDN requires a powerful southbound interface to manipulate the basic functions of hardware in the forwarding plane.

OpenFlow [2] is widely cognized as an early SDN implementation and also a popular southbound interface of SDN, which derives from the "clean-slate" project of Stanford. Supported by open networking foundation (ONF), the version of OpenFlow has been continuously updated, and in a recent version OpenFlow1.3 [3], 40 match fields are defined, which take up a total storage space of up to 1227 bits. Though increased match fields have broadened the application scope of OpenFlow, the pressure of storage space in ternary content addressable memory (TCAM) sets a great limit on the size of flow table. What's worse, TCAM has a disadvantage of high cost-to-density ratio (about US$350 for a 1 M-bit chip) and high power consumption (about 15 Watt/Mbit), which hinders greatly the expansion of TCAM size in commodity devices.

To store more flow entries in a TCAM of a practical size, compression of flow table is thus considered. The flow table compression of OpenFlow differs from other compressions such as IP routing table and ACL table in two aspects:

- *One flow entry may consist of many fields while some of them are redundant for a certain data flow.* For example, in OpenFlow1.3, a flow entry may contain both TCP source/destination address and UDP source/destination address, while in practice, TCP and UDP are two different layer4 protocol and would not exist in the same data flow.
- *Each field of a flow entry has a probability of being be masked in a flow table.* In traditional IP routing table, wildcard matching has been long used and the longest prefix matching policy of IP determines that the masked bits in an IP address must be consecutive and at the end of the address. However, in OpenFlow, arbitrary fields could be masked, so the possible position of masked bits spread all across the flow entry and are under no limitation of consecution.

The stated reasons make flow table compression of OpenFlow somehow different from previous works. Based on the problems stated above, this paper makes contributions in following aspects:

(1) A field structure analysis model of OpenFlow is presented. In OpenFlow1.3, up to 40 fields are defined, while a large increase in the number of fields don't necessarily mean a large increase of entry amount. In fact, there are some special relationships among fields that could contribute to the compression, on which the model of this paper is mainly based.
(2) A pre-compression model for TCAM is proposed. According to field relationships, a mapping scheme is proposed to compress the lengths of some fields before sent to TCAM

Manuscript received July 15, 2020; revised October 9, 2020; approved for publication by Jin-Ghoo Choi, Division 3 Editor, November 15, 2020.

This work was supported in part by The National Key Research and Development Program of China (No. 2020YFB1804803), The National Nature Science Foundation of China (No. 61521003, No. 61872382, No.61902361), The Research and Development Program in Key Areas of Guangdong Province (No. 2018B010113001), and The Fundamental Research Project of Shenzhen Municipality (JCYJ20170817115335418).

C. Zhang is with the National Digital Switching System Engineering and Technological R & D Center, Zhengzhou, China, and School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, China, email: 2000007@zzuli.edu.cn.

P. Sun is with the National Digital Switching System Engineering and Technological R & D Center, Zhengzhou, China. email: sphshine@126.com.

G. Hu is with the School of Computer Science, Shenzhen Institute of Information Technology, Shenzhen, China. email: hugw@sziit.edu.cn.

L. Zhu is with the School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, China, email: lzhu@zzuli.edu.cn.

G. Hu is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2020.000033

for match, thus reducing the cost of one flow entry in TCAM. Besides, such compression allows for unforeseen circumstances of the addition of new field value, which ensures the incremental capability of this scheme.

(3) A simulation experiment is carried out, which gives a clear view of the performance. We obtain flow tables in test networks of OpenFlow and carried out our compression algorithm. The results show that our compression ratio reaches more than 60%.

The rest of the paper is organized as follows: Section II describes the related work. Section III gives basic analyses of the features of OpenFlow flow tables and presents a mathematical model as the direction of compression. Section IV illustrates our compression analysis with a demonstration algorithm, and describes the details of compression scheme. Section V describes the hardware support for our scheme. Section VI validates the performance of our compression model with a simulation experiment. At last, a conclusion of this paper is presented.

II. RELATED WORK

Match of OpenFlow tables has similarity with traditional packet classification. The flexibility and programmability of OpenFlow in forwarding plane is partly based on the number of fields it supports. However, expansion of field number also brings the problem of flow table explosion, adding pressure to both physical equipment storage and controller maintenance [4].

Table compression has already been a hot topic since the large scale deployment of IP networks. In IP networks, packet classification also aims at multi-fields in a packet header, which is similar with OpenFlow. Existing studies about packet classification could help in the research of flow table compression in OpenFlow networks. Recursive flow classification (RFC) [5] aims at the lookup speed of packet classification and achieves high throughput, but the high cost of storage space and low table update speed set a limit on the table scale. FRFC [6] improves the update speed of RFC, but still costly in storage space. Grid of Tie [7] focuses on the refinement of search tree in packet classification, which achieves low storage space, but the complexity of tree structure hinders its speed of incremental update. HiCuts [8] and HyperCuts [9] focuses mainly on low-dimensional fields, while as the number of field dimension rises, the problem of storage space and lookup efficiency is exposed.

In traditional routing table and access control list (ACL), some match policies are similar to that of OpenFlow. Zeng and Yang [10] studies the optimization problem of ACL, which employs cross-coverage and inclusion relationships to reduce the number of ACL. ACL compressor [11] illustrates a high compression ratio model in their paper, which achieves the compression ratio of 50.22% by a divide-and-conquer approach. Karpilovsky *et al.* [12] propose to deploy a management system to reduce the router state by 70%. The decomposition and recombination process perform well in multi-fields compression, but such approach doesn't apply to arbitrary masking of fields in OpenFlow. Orange [13] reduces the storage space of range-based flow table with a new look-up process in TCAM. Rottenstreich et al. [14] also address the range-based storage problem by changing the looking-up process. Sun *et al.* [15] change the encoding method of range-based tables in the TCAM and increase the compressing efficiency of ranges. Li *et al.* [16], [17] propose to use pre-classifier to reduce the storage space of ranges in traditional routing table, which proves to work well. Some works also concentrate on manipulating the storage space of the flow table based on the consideration of the whole network functions [4], [18]–[20], but these schemes cannot be flexibly used without a full analysis of the network functions such as routing. There are also many schemes that employ decision trees for packet classification. For example, PartitionSort [21], CutSplit [22], TabTree [23], NeuroCuts [24], CutTSS [25], and NuevoMatch [26] all proposed a certain kind of method to construct a decision tree for fast packet classification. These schemes mainly aim at alleviating the search complexity of the packet classification in with the rules stored in RAM.

Due to the requirement of arbitrary mask in fields, lookup process in OpenFlow relies heavily on TCAM for a high performance, which in practice sets a great limit on the table scale of OpenFlow. Therefore, the flow table storage problem is more serious than that of traditional routing table as aforementioned. Curtis *et al.* [27] pointed out the difference between the flow table of OpenFlow and traditional routing table, and proposed DevoFlow, a model to reduce flow table size. However, its algorithm is weak in extensibility. DIFANE [28] reduces the table size kept in one switch by splitting the rule set into different switches, and ensures right match of flow entry by redirecting flow to intermediate switches. However, in a practical network, rules are not static, so frequent rule changes bring additive pressure to controller of DIFANE networks. Zhang *et al.* [29] try to redistribute the traffic to reduce the size of the flow table in each switch, but the traffic redistribution also brings many problems such as routing changes. Compact TCAM proposed by Kannan K et al. [30] discuss in detail the reliance of TCAM in OpenFlow and the drawbacks of TCAM. In this model, flow table size is reduced by adding tags to flow as a replacement of full-size entry match, but the process of adding tags in egress switches still need a full-size entry match of OpenFlow. LightFlow [31] analyses the relationship between wildcard match and exact match in flow table, based on which proposes a GPU based parallelization algorithm. Also, LightFlow is not suitable for OpenFlow in match field combination. H-SOFT [32] proposes a field partition algorithm to reduce storage space by the reduction of entry numbers in sub-tables. However, such partition process loses the support of arbitrary mask in flow table, thus damaging the fine-grained control ability of OpenFlow.

It should also be noted that like OpenFlow also contains the fields whose value could be range value (such as source port and destination port). The range expansion problem introduced by such range field brings much pressure on the storage space of TCAM. Schemes such as DIPRE [34] and LIC [35] aim at alleviating this problem with unused bits in every TCAM entry, which achieve good performance but rely on enough extra bits. Therefore, compressing the bit-width of OpenFlow entry also helps in the reduction of range expansion problem.

III. MATHEMATICAL MODEL

In this section, a mathematical model of compression is proposed, with OpenFlow1.3 considered as an example to execute flow table compression algorithm on. Since flow entry structure is one important element in our compression model, the analysis of structure is firstly presented.

The aim of flow table compression is the contraction of flow entry size while at the same time maintain the intact function of the flow entry. Basically, there are two kinds of storage medium for flow tables, one being RAM and the other TCAM. As stated in Section I, TCAM has the advantage of high match speed and arbitrary wildcard match support, but a notoriety of the high power consumption and hardware cost. On the contrary, RAM based storage is much more economical, but inefficient in search speed and helpless in arbitrary wildcard match. This model is just based on such relationship between TCAM and RAM, which aims to maximize the flow table storage ability of a hardware device with limited hardware resources.

Generally, when faced with storage pressure of TCAM, RAM has always been resorted to for an alternative. In RAM, the search process of multi-field table is usually based on trie or tree structure, but when it comes to OpenFlow, such structures are helpless, because in OpenFlow, arbitrary mask of fields are supported. In traditional multi-field match table, since wildcard is not arbitrary, redundant tree or trie nodes could be removed, thus the total amount of nodes are just of the same order of magnitude as the amount of entries. However, with the employment of arbitrary wildcard, redundant nodes might also be considered as nodes with masked value, thus should be reserved, as a consequence of which the problem of storage space explosion is inevitable.

To solve this problem, a model of RAM pre-compression is introduced, together with corresponding algorithm for the optimization problem. In this model, packet header is first partitioned into sub-sets, and then sent to RAM for the first step of compression. After the pre-compression algorithms processed in RAM, a new format of packet header with reduced bit length is then sent to TCAM for final match. Since RAM is also a limited resource, this model also takes into consideration of the tradeoff between RAM and TCAM.

The basis of flow table compression is the fact that a certain field of a protocol is usually designed to allow for increment. Take ETH_TYPE into example, the 16-bit field could accommodate up to 65536 type codes. However, in practice, only hundreds of them are assigned, among which only dozens are in common use. There are also other cases such as IP address. The 32-bit address length defines a space of a total amount of 2^{32} IP addresses, while in practice, a certain router or switch should only handle a small portion of them, say, 100 thousand. Similarly, fields like ETH_SRC, ETH_DST, ETH_TYPE are also redundant in field length considering the de facto values a certain hardware would encounter. Therefore, we can take advantage of such property of header fields to carry out the table compression.

Definition 1: Orthogonal fields pair P_O, which represents such two fields that are collided to each other and could not exist in a packet header at the same time. In P_O, if one field exists, it's certain that the other would not exist, and vice versa. For example, <TCP_SRC, UDP_SRC> is a P_O, and <TCP_DST, UDP_DST> is also a P_O.

$$\Pi = \begin{bmatrix} \ddots & & \\ & R_{ij} & \\ & & \ddots \end{bmatrix} \left(i,j \in [1,13]\right)$$

Relationship	(i,j)
$R_{i,j} = R_C$	$\forall i = j(2-4,2-4),(6,7)$ $(7,6),(8,9),(9,8),(10,11)$ $(11,10),(12,13),(13,12)$
$R_{i,j} = R_O$	$(6,9),(7,8),(8,7),(9,6),$ $(10,12),(10,13),(11,12),$ $(11,13),(12,10),(12,11),$ $(13,10),(13,11)$
$R_{i,j} = R_E$	$(6,8),(7,9),(8,6),(9,7)$

Fig. 1. Relationship matrix of 13 required fields for OpenFlow 1.3.

Definition 2: Evolution fields pair P_E, which represents two fields that one is the updated version of the other. P_E shows a characteristic of evolution, which means the former field possesses all the functions that the latter one has. For example, <IPv4_SRC, IPv6_SRC> is a P_E. P_E is critical in field mapping and also helps dealing with flow header partition illustrated in the following sections.

Definition 3: Coexisting fields pair P_C, which represents two fields that one field exists if and only if the other exists. P_C shows a relationship of dependence. Therefore, during the field partition process, P_C should be taken into consideration to improve the algorithm efficiency.

Definition 4: Storage coefficient η. The reduction of TCAM is to some degree at the cost of the increase in RAM consumption. η is the reflection of such condition, which by definition is the reduction of TCAM in bit at per bit consumption of RAM. In practice, η could help accommodating this algorithm in hardware devices with different size of TCAM and RAM.

A matrix of field relationship is shown in Fig. 1 as an example. The example of relationship matrix Π is based on common sense of field relationship of the 13 required fields in OpenFlow v1.3. In Fig. 1, the indicator i and j denotes the number of field in the 13 match fields of OpenFlow v1.3. Specifically, the table in Fig. 1 gives a detailed description of the relationship type of each match field, e.g., in the relationship $R_{i,j} = R_E$, the numbers on the right column of the table denote that (R_6, R_8), (R_7, R_9), (R_8, R_6) and (R_9, R_7) belong to the relationship R_E. It should be mentioned that some fields may have more than one relationship type, in which case we heuristically sort the priority of relationship as $P_E > P_O > P_C$ according to the compression ratio listed in the following sections. In a certain deployment environment, Π may change, for example, the number of PE may increase.

Table 1 shows the variables used in this model.

Table 1. Variable notation for RETCAM model.

Notation	Representation of the symbol or symbol
F	Sets of all different fields in this model
T	Flow table
M	Total number of entries in T
N	Total number of fields in F
N_i	Number of Fields in subset i $(1 \leq i \leq k)$
M_i	Number of entries in subset i $(1 \leq i \leq k)$
l_i	Bit width of field i $(1 \leq i \leq k)$
$l_{mn}{}'$	Bit width of compressed pair P_O
f_n	Field n $(1 \leq n \leq N)$
w_i	Bit width of subset i in RAM $(1 \leq i \leq k)$
$w_i{}'$	Bit width of search output of subset i $(1 \leq i \leq k)$

A. Field Partition

As mentioned above, almost all protocols contain redundancy in field length. On the one hand, redundancy is necessary for future development of such protocol, since in time scope, new version of the protocol may introduce new type field, and in user scope, addition of new users may bring more addresses into use; on the other hand, since every field allows for its own redundancy, the overall redundancy in a group of fields is just too large a waste of storage space. Therefore, we can just allot some redundancy to two or more field together, so the overall redundancy is reduced while at the same time there is enough space for such group of fields to extend their application scope.

In the compression algorithm, different fields should be treated in different ways, with the implementation in RAM. Therefore, partition of fields into sub-sets plays an important role in the reduction of RAM space, which both reduce entry numbers of sub-sets and increase the pre-compression speed with parallel operation. Let $F = \{f_n | n = 1, 2, \cdots, N\}$ be the fields set of all N fields in the flow table.

First, partition F into k sub-sets according to P_O and P_C defined above, of which the exact value of k is acquired by the equation set listed below, and build a sub-table for each sub-set. For each field, whether it should be assigned to a sub-set is also limited by coefficient η, which balances the reduction in TCAM size and the cost in RAM space. In the ith sub-set, the number of fields is N_i, which satisfies (1).

$$N{=}N_1{+}N_2{+}\cdots + N_k. \tag{1}$$

In each sub-set, since it only covers the information of the fields within the sub-set, entry number should be less than M. Let the entry number of each sub-set be M_i.

B. Inter-field Merge

Inter-field merge applies to P_O and P_C. Within each sub-set, if there exist $f_m, f_n \in F_i$ that $< f_m, f_n >$ is a P_O or P_C, we can carry out the compression process. Let l_m be the bit length of f_m and l_n be the bit length of f_n, then before compression, the storage space for these two fields is $(l_m + l_n)$ bits. In practice, the amount of values of f_m and f_n in use are much less than $2^{(l_m+l_n)}$, as explained before. Therefore, we could allot a new field of length $l_{mn}{}'$, which could accommodate all the values of f_m and f_n while at the same time reserve enough space for future extension. As a result, two fields f_m and f_n could be replaced by one field, thus the total storage space is reduced. Besides, the compression ratio C_o of this P_O is expressed in (2).

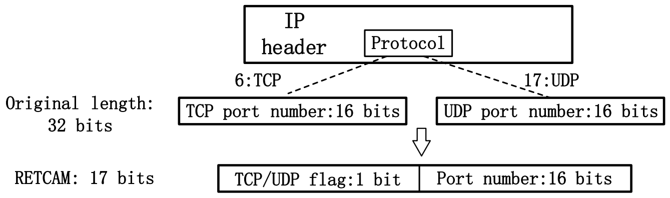


Fig. 2. An example of inter-field merge.

$$C_O = \sum_{i=1}^{M_i} l'_{mn} / (\sum l_m + \sum l_n). \tag{2}$$

The storage coefficient is given in (3).

$$\eta_{Po} = \sum_{i=1}^{M'} (l_m + l_n - l_{mn}{}') / (\sum l_m + \sum l_n). \tag{3}$$

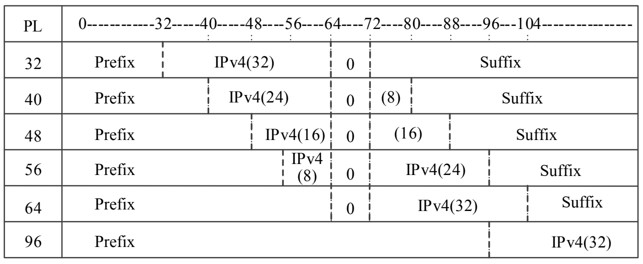
Compression of P_O or P_C doesn't affect the final match result of a packet header, of which the reasons are explained as follows. In TCAM, the entry structure is reconstructed according to compression format, for example, two fields are merged into one field. Since such operation mainly affect the compressed fields of P_O, we can classify the corresponding circumstances into such two types:

(1) Between the two orthogonal fields f_m and f_n, an entry has only one exact field value while the other field masked. In this case, such entry would be assigned with an exact match value of merged field P_O in TCAM. We now analyze this circumstance with the assumption that field f_m is with an exact value A in this entry while f_n is masked (analysis of the opposite is just the same). In practice, a coming packet header may come with a field f_m or otherwise f_n(orthogonal fields would not exist together). For different field values of f_m and f_n, search outcome from RAM would be different, so only a packet header with a value A could be matched against such entry. Therefore, collision would not happen after the merging of f_m and f_n.
(2) Both f_m and f_n are masked in the entry before compression. In this case, such entry after compression would have the merged field P_O masked. It's clear that for this circumstance, compression operation doesn't change the match outcome in TCAM.

Fig. 2 shows an example of the inter-field merge. As shown in the figure, in an original OpenFlow flow table, it takes 32 bits to store both a TCP and a UDP port number. In RETCAM, there is one flag bit to denote whether the coming packet is a TCP or a UDP packet, and the space to store the port number can be multiplexed for these two protocols. As a result, the storage width is reduced from 32 bits to 17 bits.

C. Field Mapping

Among the fields of OpenFlow1.3, there are another special relationship of fields, in which one field f_n is just the upgraded



PL	0	32	40	48	56	64	72	80	88	96	104
32	Prefix	IPv4(32)				0	Suffix				
40	Prefix		IPv4(24)			0	(8)	Suffix			
48	Prefix			IPv4(16)		0	(16)		Suffix		
56	Prefix				IPv4 (8)	0	IPv4(24)			Suffix	
64	Prefix					0	IPv4(32)				Suffix
96	Prefix									IPv4(32)	

Fig. 3. Address conversion table.

version of f_m, such as source and destination address of IPv6 versus source and destination address of IPv4.

In IPv6, source address and destination address are expanded to 128 bits, which defines enough address space for hosts. In fact, IPv6 has almost all the functions IPv4 has in network layer, together with the expansion of address space and extension of support for some special applications. Therefore, IPv6 could perform all the function of IPv4. Temporarily, smooth transition from IPv4 to IPv6 is proposed, so IPv6 and IPv4 coexist in the network.

Among all the smooth transition technologies from IPv4 to IPv6, protocol conversion technology is a popular one. RFC 6052 [33] specifies the technology of address conversion from IPv4 to IPv6, such as shown in Fig. 3. As shown in Fig. 3, there are six types of mapping method from IPv4 to IPv6, which have different prefix lengths (PL). In practice, the network has to adopt one of the mapping method as the mapping format from IPv4 to IPv6. For example, if we adopt the mapping method with the PL value 40, then the first 24 bits of the IPv4 address is mapped onto bit 40–64 of the IPv6 address and the last 8 bits of the IPv4 address is mapped onto bit 72–80 of the IPv6 address. The prefix and suffix value can be assigned to 0.

In TCAM, only the total length of prefix is concerned. After conversion from IPv4 address to IPv6 address, the masked bits of original IPv4 fields in an entry are still masked in TCAM. RFC 6052 suggests six different prefix length, among which an appropriate one for this model should be considered. In the IPv4-converted IPv6 address, new prefix in a flow entry is the combination of original prefix in IPv4 address and the newly-added prefix.

The compression ratio in field mapping is:

$$C_E = \sum_{f_n, f_m} l_n / \sum_{f_m, f_n} (l_m + l_n). \tag{4}$$

D. Intra-field Compression

For other fields, there could also be RAM-based search algorithms to reduce the field length. For example, the 48-bit fields ETH_SRC and ETH_DST could also be compressed. In this way, hash algorithm could be used to accelerate the compression, so collision should be carefully processed to ensure a unique outcome for every income value. Generally, we define the bit-length of sub-set i in RAM with $w_i (1 \leq i \leq k)$, of which w_i is the sum of both original field length and compressed field length. Besides, the bit length of match outcome in RAM which is sent to TCAM is defined with $w_i{}'$. By field partition, repetitive values of a field in a total flow table could be merged into one in a sub-set table, thus reducing the entry amount. Since the overall entry amount is denoted by M as mentioned above, we can denote the entry amount of sub-set with M_i. Therefore, the compression ratio C_i of sub-set i is $C_i = (M_i \times w_i)/(M \times l_i)$, and the storage coefficient η is $\eta_i = (w_i - w_i')/w_i$. Fig. 4 shows a simple example of the intra-field compression, which is a hash of the MAC address. With the has operation on the MAC address, the width can be reduced from 48 bits to 24 bits.

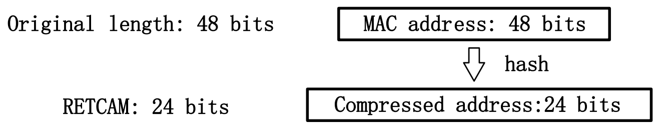


Fig. 4. An example of intra-field merge.

Algorithm 1 Sub-sets creation

```
    Input: match field set $F$, the relationship among fields,
    their compression ratio $\eta$ and the threshold $\eta_{TH}$
    Output: sub-sets of match fields
 1: Begin
 2:  sort relationship pairs in $\Pi$ in descending order of $\eta$;
 3:  While $F$ is not empty
 4:    extract the relationships $R$ in $F$ with the maximum
       value $\eta$;
 5:    If $(\eta_R \leq \eta_{TH})$
 6:      generate sub-set $s_R$;
 7:      and remove used match fields and update $F$;
 8:  For $f_i \in F$
 9:    If $(\eta_i \leq \eta_{TH})$
10:      generate sub-set $s_i$;
```

IV. ALGORITHMS

In this section, an algorithm as an implementation of the model is presented. The overall algorithm contains mainly three stages, which is sub-sets creation, pre-compression in sub-sets and recomposing of header field. Algorithm1 and Algorithm 2 show the process in detail.

Algorithm 1 partitions the overall fields in F into several sub-sets based on P_O, P_E, and P_C in relationship matrix Π, on which three types of pre-compression algorithm could be performed. Creation of a sub-set is according to the storage coefficient η, by which the user could regulate the complexity of RETCAM with threshold value η_{TH} according to the hardware condition.

Algorithm 2 accommodates the original flow table into corresponding sub-flow tables created according to sub-sets. During the accommodation process, field value of each low entry is inserted into the tree as a node or the hash table. For fields like protocol number, values are usually consecutive, thus searching process of such type is based on binary search.

Algorithm 2 Sub-sets compression

```
     Input: sub-sets of fields, original flow table $T$
     Output: compressed sub-flow tables
  1: Begin
  2:   For sub-set $s_i$ in sub-sets
  3:     select the compression type of $s_i$(hash or linear);
  4:     For j = 1 to M (M is the total number of flow entries)
  5:       If (hash)
  6:         get the hash value $e_{i,j}$ of $s_i$;
  7:         update $e_{i,j}$ into the jth flow entry;
  8:       Else if (linear)
  9:         get the hash value $e_{i,j}$ of $s_i$;
 10:         update $e_{i,j}$ into the jth flow entry;
```

V. HARDWARE SUPPORT

Like all other TCAM storage encoding schemes as mentioned in Section III, RETCAM also relies on a minor modification of the hardware in dataplane to conduct the functions, which is shown in Fig. 5. When a packet comes, it's firstly processed in the packet processor. In an ordinary dataplane device, after the packet header is extracted from the packet, the header field is reformatted as the match field for matching. In the modified circuit of RETCAM, the header field is then processed by the pre-encoding circuit. As mentioned in this section, RETCAM uses a combination of SRAM and TCAM to decide the match outcome. Since the price is SRAM is omittable compared to TCAM, the table stored in SRAM takes a low cost There are also interfaces to the control plane as shown in Fig. 5, which is saved for the updating of the encoding scheme. Also, the TCAM space is divided into two parts with different widths: One part is for the storage of the RETCAM-compressed flow table, which takes most of the TCAM storage space; the other part is for the original flow table, which takes only a small proportion of the TCAM space. The reason is explained below.

The update of the flow table can be flexibly carried out with this architecture, which is shown in Fig. 6. First, for deleting a flow entry, it is the same for a RETCAM-compressed flow table and ordinary flow table, i.e., just deleting an entry from the flow table in TCAM. Then, for adding a new flow entry, we first calculate the compression algorithms of RETCAM for this flow entry and see after the compression, if this flow entry is in conflict with any flow entries in the existing flow table (i.e., two different flow entries have the same compression outcome). If there is no conflict, then we just add this compressed new flow entry into the flow table, which is also the same as adding a flow entry to an ordinary flow table. For the flow entry that is in conflict with the existing flow table (which will only happen in the hash operation of the MAC address and has a small probability), we store this flow entry in a small scale full-width TCAM space in the switch. When the small scale full-width TCAM space is nearly full, we take all the flow entries in a whole and run the compression algorithm in the SDN controller again to get a new compressed flow table, which will happen much less frequent than the update of the flow table.

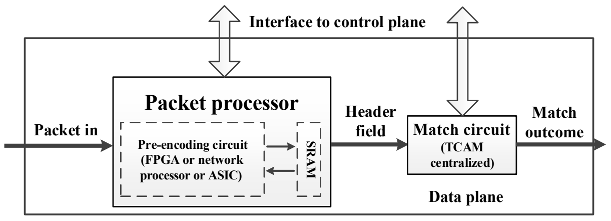

Interface to control plane
Packet in
Packet processor
Pre-encoding circuit (FPGA or network processor or ASIC)
SRAM
Header field
Match circuit (TCAM centralized)
Match outcome
Data plane

Fig. 5. Hardware implementation of RETCAM.

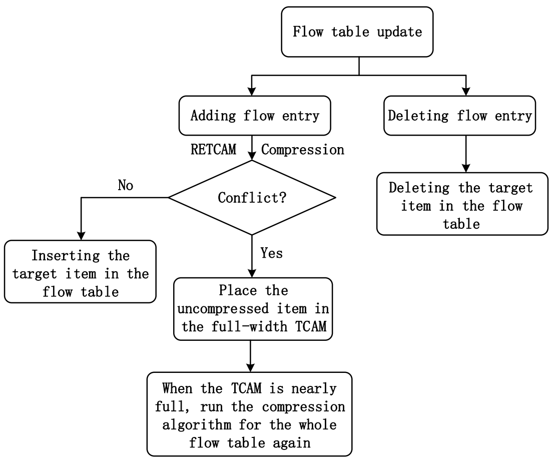

Flow table update
Adding flow entry
Deleting flow entry
RETCAM Compression
Conflict?
No
Yes
Inserting the target item in the flow table
Place the uncompressed item in the full-width TCAM
When the TCAM is nearly full, run the compression algorithm for the whole flow table again
Deleting the target item in the flow table

Fig. 6. Flow table update process.

VI. SIMULATION AND EVALUATION

Since OpenFlow has not been publicly used in networks, it's infeasible to carry out the evaluation with real data in large scale OpenFlow networks. The flow table in this experiment is collected from the test network of national engineering research center for broadband networks & applications, which roughly reflects the flow table condition in a practical OpenFlow network. The algorithms of RETCAM are simulated in Matlab 2012b, which is operated on a Core i7-4790 CPU with 8 GB memory. Under this experiment environment, we compare the performance of RETCAM with the following baselines:

H-SOFT [32] proposes to partition the flow table into subtables according to the match fields in use. With this partition, the number of match field in each sub-table can be reduced. However, this method relies heavily on the pre-analysis of the flow table, which may fail when new flow entries come.

RFC [15] propose to change the encoding method of ranges in flow tables. A header field to be matched will firstly be processed in a FPGA-based programmable hardware to encode the range information. In this way, the storage space of ranges in flow tables can be greatly reduced. However, this method only considers the compression of ranges in flow tables.

SplitIP [17] propose a splitting algorithm to separate the prefix-based routing table into TCAM blocks to reduce the TCAM consumption. With the table separation, a preclassification operation should be performed for prefix looking up. However, this scheme only considers the storage consumption of prefixes.

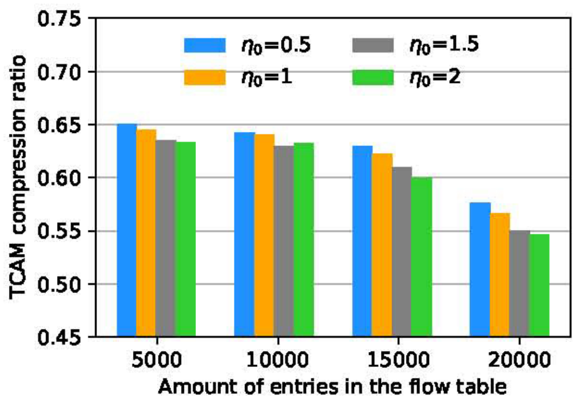


Fig. 7. Compression ratio of RETCAM.

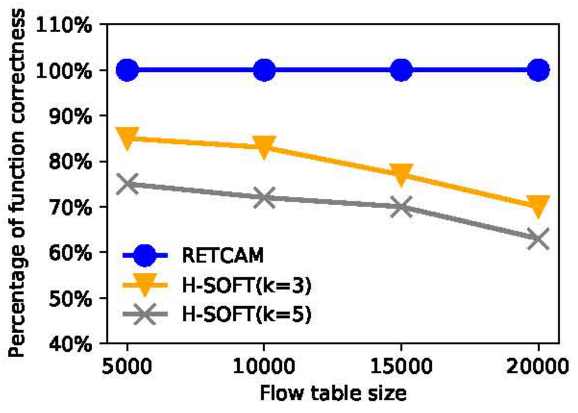


Fig. 8. Comparison of function integrity among different schemes.

A. TCAM Reduction of RETCAM

RETCAM aims to reduce TCAM consumption with pre-compression scheme, which reduces the entry size in the preprocessing of RAM. Taking into consideration of different physical truth of hardware, RETCAM is self-adaptive in different application environment. Fig. 7 shows the value of performance index ρ against different flow table sizes and different value of storage coefficient η, which is on the condition that flow partition number k is 9. From Fig. 7, we can find that compression ratio ρ increases with the decrease of storage coefficient η, while as flow table size grows, compression ratio ρ changes slowly. For example, compression ratio ρ remains around 0.6 as flow table size grows from 5000 to 25000. This result validates the incremental stability of TCAM reduction in RETCAM, which is an important characteristic in practice.

B. Function Integrity Comparison between RETCAM and H-SOFT

One great advantage of OpenFlow is the fine-grained control ability of data flow. According to the OpenFlow whitepaper, arbitrary combination of fields in an entry is supported, which would cater to many special user needs. However, existing routing tables in current networks usually focus on one or several fields that are widely adopted for routing. Therefore, format conversion alone from traditional networks to OpenFlow just like H-SOFT did could not fully reflect the flow control need of OpenFlow networks, which omits many possible combinations of fields in OpenFlow that do not exist in traditional networks.

In H-SOFT, fields are partitioned into many field-groups, and match process is operated by one choice of such group. Even though by careful identification of field relationship such as conflict and coexistence, group partition could avoid damage of function integrity to some extent, the large diversity of match field combination in OpenFlow could not support so many group partitions as H-SOFT did. Merely format conversion from traditional routing table to OpenFlow format just cover up the fact that the divide-and conquer policy in H-SOFT damages the function integrity of OpenFlow, since the diversity of match field combination in traditional routing table is fixed to a very small scope.

In the real flow table of OpenFlow collected from RETCAM, the diverse combinations of match fields roughly reflect the flexibility of OpenFlow. Under this circumstance, the damage to function integrity of H-SOFT is exposed. In our simulation of H-SOFT, the number of malfunction cases suffer from a manifest increase as the size of flow table grows, while RETCAM performs no functional error in this experiment. Fig. 8 shows the percentage of flow entries that could be correctly functioned in different schemes. When subset number k grows larger, the percentage of flow entries that could be correctly functioned drops significantly in H-SOFT, thus we gave up the simulation when $k > 5$ in H-SOFT, since there are too many malfunction cases. As shown in the figure, RETCAM can maintain all the function integrity of the flow table for OpenFlow. Specifically, since RFC and SplitIP do not change the information contained in the flow table (which means a full function integrity), we do not add them in the analysis of the function integrity part.

C. Compression Comparison between H-SOFT and RETCAM

H-SOFT proposed a flow table compression algorithm based on the principle of divide-and-conquer. By the partition of sub-tables, one flow may choose one of them to carry out the match process. However, when the fine-grained flow control ability of OpenFlow is taken fully used of, arbitrary wildcard may occur in a flow table, which would greatly hinder the performance of H-SOFT. One premise of the divide-and-conquer principle is the classification of match types in a flow table, while in OpenFlow the distribution of match field in a flow entry may sometimes be too casual to categorize. RFC and SlpitIP concentrate on the efficient storage of range-based match fields such as TCP port range and IP prefixes. Therefore, such two schemes do not bring much benefit in the reduction of match fields in OpenFlow. As shown in the figure, under different flow table sizes, the compression ratios of RFC and SplitIP are stable, which is around 0.17 and 0.28, respectively. In RETCAM, since the pre-compression outcomes in different sub-tables are recomposed into one summarizing header field, support of arbitrary wildcard in OpenFlow is not influenced. Fig. 9 shows the comparison of performance among different schemes, where the threshold n in H-SOFT set to 10000. As Fig. 9 shows, RETCAM performs

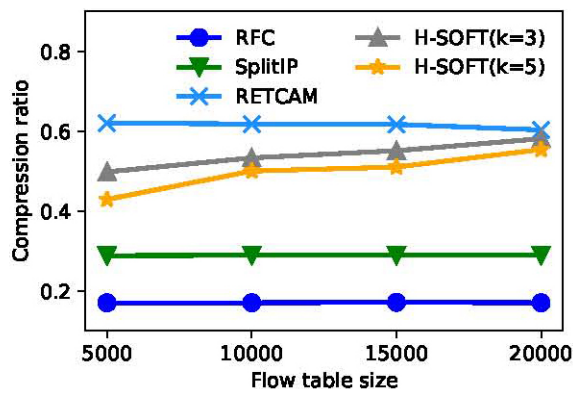


Fig. 9. Comparison of compression ratio among different schemes.

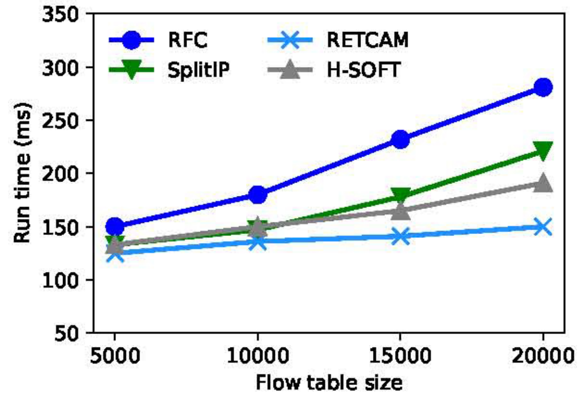


Fig. 10. Run time comparison of different schemes.

the best among all schemes. Besides, RETCAM shows a bigger advantage when flow table size is small.

D. Computation Time Comparison

We compare the computation time of different schemes under our test environment, of which the result is shown in Fig. 10. Since RFC and SplitIP rely on complicated relationship analysis among different flow entries and match fields, such two schemes have a much higher computation complexity when the scale of the table size grows.

VII. CONCLUSION

Flow table of OpenFlow requires much storage space of TCAM. This paper proposed an efficient TCAM saving model for an OpenFlow switch based on the analysis of relationships between different fields. By the pre-compression procedure of inter-field compression, field mapping and intra-field compression, RETCAM effectively reduces the table size in TCAM as much as 60%, thus alleviating the pressure of storage in TCAM. With a modest run time of thousands of entries per second and a good performance of incremental deployment. In a word, RETCAM can efficiently reduce the TCAM consumption in the storage of OpenFlow tables.

REFERENCES

[1] J. Montag, “Software defined networking mit OpenFlow,” *in Proc. IITM*, 2013.

[2] N. McKeown *et al.*, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Comput. Commun. Review*, vol. 38, no. 2 pp. 69–74, Mar. 2008.

[3] OpenFlow 1.3.0 specification. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/specification/openflowspec-v1.3.0.pdf.

[4] Z. Guo *et al.*, “STAR: Preventing flow-table overflow in software-defined networks,” *Comput. Networks* vol. 125, pp. 15–25, Oct. 2017.

[5] J. Cao and B. Chen, “Memory-optimized RFC packet classification algorithm merge RFC,” *J. Chinese Comput. Syst.*, vol. 33, no. 4, pp. 865–868, Mar. 2012.

[6] W. Pak and S. Bahk. “FRFC: Fast table building algorithm for recursive flow classification,” *IEEE Commun. Lett.*, vol. 14, no. 12, pp. 1182–1184, Dec. 2010.

[7] H. Lim, S. Lee, and E. E. Swartzlander, “A new hierarchical packet classification algorithm,” *Comput. Networks*, vol. 56, no. 13, pp. 3010–3022, Sept. 2012.

[8] P. Gupta and N. McKeown, “Packet classification using hierarchical intelligent cuttings,” *IEEE Hot Interconnects*, vol. 40, 1999.

[9] S. Singh, F. Baboescu, G. Varghese, and J. Wang, “Packet classification using multidimensional cutting,” *in Proc. ACM SIGCOMM*, Aug. 2003.

[10] K. Y. Zeng and J. H. Yang, “Towards the optimization of access control list,” *J. Software*, vol. 18, no. 4, pp. 978–986, Apr. 2007.

[11] A. X. Liu, E. Torng, and C. R. Meiners, “Compressing network access control lists,” *IEEE Trans. Parallel Distributed Syst.*, vol. 22, no. 12, pp. 1969–1977, Dec. 2011.

[12] E. Karpilovsky, M. Caesar, J. Rexford, A. Shaikh, and J. van der Merwe, “Practical network-wide compression of IP routing tables,” *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 4, pp. 446–458, Dec. 2012.

[13] L. Schiff, Y. Afek, and A. Bremler-Barr. “Orange: Multi field openflow based range classifier,” *in Proc. ACM/IEEE ANCS*, May 2015.

[14] O. Rottenstreich *et al.*, “Optimal in/out TCAM encodings of ranges,” *IEEE/ACM Trans. Netw.* vol. 24, no. 1, pp. 555–568, Feb. 2016.

[15] P. Sun, J. Lan, P. Wang, and T. Ma, “RFC: Range feature code for TCAM-based packet classification,” *Comput. Networks*, vol. 118, pp. 54–61, May 2017.

[16] W. Li, X. Liu, W. Le, H. Li, and H. Zhang, “A practical range encoding scheme for TCAMs,” *in Proc. ACM SIGCOMM Posters and Demos*, Aug. 2019.

[17] L. Wenjun *et al.*, “A power-saving pre-classifier for TCAM-based IP lookup,” *Comput. Networks*, vol. 164, p. 106898, Dec. 2019.

[18] Z. Guo *et al.*, “Balancing flow table occupancy and link utilization in software-defined networks,” *Future Generation Comput. Syst.* vol. 89, pp. 213–223, Dec. 2018.

[19] Z. Guo, S. Hui, Y. Xu, and H. J. Chao, “Dynamic flow scheduling for power-efficient data center networks,” *in Proc. IEEE/ACM IWQoS*, June 2016.

[20] Z. Guo *et al.*, “JumpFlow: Reducing flow table usage in software-defined networks,” *Comput. Networks*, vol. 92, pp. 300–315, Dec. 2015.

[21] S. Yingchareonthawornchai, J. Daly, A. X. Liu, and E. Torng “A sorted-partitioning approach to fast and scalable dynamic packet classification,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1907–1920, Aug. 2018.

[22] W. Li, X. Li, H. Li, and G. Xie, “CutSplit: A decision-tree combining cutting and splitting for scalable packet classification,” *in Proc. IEEE INFOCOM*, Apr. 2018.

[23] W. Li, T. Yang, Y.-K. Chang, T. Li, and H. Li, “TabTree: A TSS-assisted bit-selecting tree scheme for packet classification with balanced rule mapping,” *in Proc. ACM/IEEE ANCS*, Sept. 2019.

[24] E. Liang *et al.*, “Neural packet classification,” *in Proc. ACM SIGCOMM*, Aug. 2019.

[25] W. Li *et al.*, “Tuple space assisted packet classification with high performance on both search and update,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1555–1569, July 2020.

[26] A. Rashelbach, O. Rottenstreich, and M. Silberstein, “A computational approach to packet classification,” *in Proc. ACM SIGCOMM*, Aug. 2020.

[27] A. R. Curtis *et al.,* “DevoFlow: Scaling flow management for high-performance networks,” *in Proc. ACM SIGCOMM*, Aug. 2011.

[28] Yu Minlan, J. Rexford, M. J. Freedman, and J. Wang “Scalable flow-based networking with DIFANE,” *ACM SIGCOMM Comput. Commun. Review*, vol. 40, no. 4, pp. 351–362, Aug. 2010.

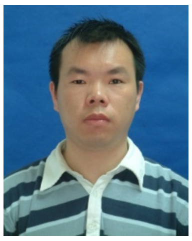
[29] S. Q. Zhang *et al.*, “TCAM space-efficient routing in a software defined network,” *Comput. Networks,* vol. 125, pp. 26–40, Oct. 2017.

[30] K. Kannan and S. Banerjee, “Compact TCAM: Flow entry compaction in TCAM for power aware SDN,” *in Proc. ICDCN*. Jan. 2013.

[31] N. Matsumoto and M. Hayashi, "LightFlow: Speeding up GPU-based flow switching and facilitating maintenance of flow table," *in Proc. IEEE HPSR*, June 2012.
[32] J. Ge, Z. Chen, Y. Wu, and Y. E "H-SOFT: A heuristic storage space optimisation algorithm for flow table of OpenFlow," *Concurrency Computation Practice Experience*, vol. 27, no. 13, pp. 3497–3509, Aug. 2015.
[33] RFC 6052. [Online]. Available: http://datatracker.ietf.org/doc/rfc6052/.
[34] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," *ACM SIGCOMM Comput. Commun. Review*, vol. 35, no. 4, pp. 193–204, Aug. 2005.
[35] A. Bremler-Barr, D. Hay, and D. Hendler. "Layered interval codes for TCAM-based classification," *in Proc. ACM SIGMETRICS*, June 2008.

Chaoqin Zhang is currently pursuing the Ph.D. degree with the National Digital Switching System Engineering and Technological R&D Center, China. He is also an Associate Professor with the Zhengzhou University of Light Industry. His research interests include Internet architecture, data mining and network security.

Penghao Sun is a Ph.D. Candidate at China National Digital Switching System Engineering & Technological R&D Center. His current research interests are in SDN, networking algorithms and packet classification.



Guangwu Hu received the Ph.D. degree in Computer Science and Technology from Tsinghua University in 2014. Then he became a Post-Doctoral with the Graduate School at Shenzhen, Tsinghua University. He is currently an Associate Professor with the Shenzhen Institute of Information Technology. His research interests include software-defined networking, next-generation Internet and Internet security.

Liang Zhu received Ph.D. degree in Computer Science and Technology from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in October 2017. He is currently a Lecturer with the Institute of Computer and Communication Engineering at Zhengzhou University of Light Industry, Henan, China. His current research interests include mobile social networks, personalized service recommendation, and privacy preserving.