Fast TCAM-Based Multi-Match Packet Classification Using Discriminators

Hsin-Tsung Lin and Pi-Chung Wang, *Member, IEEE*

Abstract—Ternary content addressable memory (TCAM) is a widely used technology for network devices to perform packet classification. TCAM compares a search key with all ternary entries in parallel to yield the first matching entry. To generate all matching entries, either storage or speed penalty is inevitable. Because of the inherit disadvantages of TCAM, including power hungry and limited capacity, the feasibility of TCAM-based multi-match packet classification (TMPC) is thus debatable. Discriminators appended to each TCAM entry have been used to avoid storage penalty for TMPC. We are motivated to minimize speed penalty for TMPC with discriminators. In this paper, a novel scheme, which utilizes unused TCAM entries to accelerate the search performance, is presented. It selectively generates TCAM entries to merge overlapping match conditions so that the number of accessed TCAM entries can be significantly reduced. By limiting the number of generated TCAM entries, the storage penalty is minimized since our scheme does not need extra TCAM chips. We further present several refinements to the search procedure. The experimental results show that our scheme can drastically improve the search performance with extra 10-20 percent TCAM entries. As a result, the power consumption, which correlates to the number of accessed TCAM entries per classification, can be reduced.

Index Terms—Packet classification, TCAM, multi-match, geometric intersection

1 Introduction

Packet classification is a process employed by Internet routers to classify packets into network flows. It is applied to many Internet services such as firewall packet filtering, quality of services, and intrusion detection. Packet classification is based on rules, which consist of multiple field specifications of a packet header. The most common fields include source IP address, destination IP address, source port, destination port, and protocol. A typical rule thus has 104 bits. The specification of an IP address field is a prefix, and that of a port field is a range. The protocol field is usually an exact value. Since different services may use different fields in a packet header, a rule can ignore a field by specifying a wildcard. Each rule is associated with an action to process the matching packet, where a packet matches a rule if each field of the packet matches the corresponding field of the rule. Some network services, such as firewall and quality of services, relies on single-match packet classification, which only yields the best matching rule. The best matching rule could be the rule with the highest priority or the least-cost action. The services such as intrusion detection or multifunction devices that perform single-match packet classification for each function require multi-match packet classification to produce all matching rules.

• *The authors are with the Department of Computer Science and Engineering, National Chung Hsing University, Taichung, Taiwan 402, ROC. E-mail: aries77329@gmail.com, pcwang@nchu.edu.tw.*

Manuscript received 1 Jan. 2018; revised 17 May 2018; accepted 1 June 2018. Date of publication 15 June 2018; date of current version 29 Jan. 2019. (Corresponding author: Pi-Chung Wang.) Recommended for acceptance by R. Grant. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMSCS.2018.2847677

In the past few years, ternary content addressable memories (TCAMs) have been widely used for packet classification due to its high throughput. A TCAM entry consists of hundreds of cells, where each cell can store '0', '1', or "don't care". We denote a cell storing "don't care" as an x-bit. TCAM can thus store binary strings with arbitrary bit masks (i.e., ternary strings). TCAM compares search keys with all entries simultaneously and only needs one access to decide the first matching entry. By storing rules in a descending order of their priority, TCAM supports single-match packet classification effectively. Unfortunately, TCAM has several drawbacks, including the inefficient storage of arbitrary ranges, limited capacity, high cost and high power consumption. More specifically, the largest TCAM chip has only 72 megabits (Mb) [1]. TCAM costs about 30 times more per bit of storage than SRAM and consumes 150 times more power per bit than SRAM [2]. The extra logic and capacitive loading of TCAM also result in tripling the access time of SRAM [2]. Another research suggests that larger TCAM chips may have longer access latency [3]. All of these drawbacks directly correlate to the number of required TCAM entries.

To support multi-match packet classification, the existing algorithms suffer from either extra TCAM entries or accesses. For example, geometric intersection (GI) [4] generates one pseudo rule for each unique set of overlapping rules, where each pseudo rule stores the overlapping rules. All matching rules can thus be yielded in one TCAM access by searching for the best-matching pseudo rule. Due to the limited TCAM capacity, GI is not feasible for heavily overlapping rules. For example, our experiments show that, while applying GI to one thousand rules designed for a network intrusion detection system, more than two hundred thousand pseudo rules are generated. Another algorithm, multi-match using

discriminators (MUD) [5], does not require any redundant entries, but it may access more TCAM entries than the matching rules. In other words, a TCAM access may not always yield a successful match even there exist other matching entries. These redundant accesses also result in considerable energy wastage.

In this work, we focus on fast multi-match packet classification based on commodity TCAM. We observed that the existing algorithms have a tradeoff for either space or search efficiency. Since a commodity TCAM chip has limited space, a scheme, which can adaptively utilize unused TCAM entries to achieve better search performance, is thus preferable. We present a scheme that uses discriminators and pseudo rules simultaneously to minimize the number of redundant TCAM accesses. Our scheme selectively generates pseudo rules, which are stored in unused TCAM entries. These pseudo rules can significantly reduce the number of required discriminators. As a result, the redundant TCAM accesses are eliminated to raise search performance as well as lower power consumption. We also develop two refinements, *bypass* and *reassign*, to further improve the search performance by incorporating pre-computation information. Our experimental results show that by making use of the available TCAM entries, our scheme outperforms MUD in terms of speed performance. Both refinements provide further speed improvement. Because our scheme can effectively control the number of occupied TCAM entries, it also avoids the issue of feasibility for supporting a rule set with a large number of overlapping rules.

The rest of the paper is organized as follows. In Section 2, we describe the previous schemes for solving the problem of multi-match packet classification and some TCAM-related research. Section 3 discusses the motivation of our scheme. Section 4 introduces our scheme and two speed refinements in detail. The experimental results and discussion are described in Section 5. Finally, we conclude our work in Section 6.

2 Related Works

In this section, we first introduce some research relevant to our scheme. Then, the relationships between multi-match packet classification and the other research topics of TCAM are discussed.

2.1 Multi-Match Packet Classification

To perform multi-match packet classification without modifying the TCAM circuit, entry-invalidation exploits a valid bit, which is originally used for indicating whether a TCAM entry is occupied or not [5]. A search key is only compared with the occupied TCAM entries with an enabled valid bit. When a TCAM entry matches a search key, its valid bit is disabled so that the entry is excluded from the next TCAM access. The valid bits must be reenabled for a new search key. The major drawback of this scheme is the extra two TCAM writes for each matching entry.

Lakshminaryanan et al. propose a scheme, multi-match using discriminator, to avoid slow TCAM write operations by appending a discriminator field to each entry [5]. The discriminator field of each entry stores a sequence number, and all entries are sorted based on their sequence number in an ascending order. In the search process, a discriminator field is appended to the search key to determine the range of TCAM entries to be compared. Initially, the discriminator field is set to wildcard so that all entries are compared. When the search key matches the jth entry, the discriminator of the search key is changed to "*greater than j*" to avoid matching the jth entry again. The process repeats until there is no matching entry. Another algorithm also uses rule grouping to facilitate multi-match classification [6]. Unlike MUD, the rule grouping approach is based on the layered ranges using P^2C, a database-dependent range encoding algorithm [7]. Each rule group is identified by using one bit of a bitmap appended to each TCAM entry. The algorithm also alters the range field of a search key to shorten the bitmap length. Because the algorithm is tightly combined with range encoding, it may suffer from high update cost caused by rule insertion.

Yu and Katz proposed GI to yield all matching rules using one TCAM access [4]. Their scheme uses pseudo rules to store all rule intersections, where each pseudo rule records the identifiers of rules which match the corresponding address space of a rule intersection. Pseudo rules must be stored in front of the original rule set to ensure the search correctness. When a search key matches more than one rule, it always matches a pseudo rule to fetch the identifiers of all matching rules. Theoretically, the number of pseudo rules is $O(N^k)$, where N is the number of original rules and k is the number of fields. GI has superior speed performance, but the number of pseudo rules may be too large to be stored in a TCAM chip. To alleviate the storage penalty, set splitting algorithm (SSA) is proposed to reduce the number of pseudo rules [8]. It is based on the observation that rule intersections can be reduced by splitting a rule set into several subsets. The algorithm categorizes rules into several subsets, where the pseudo rules are generated only for the rules of the same subset. The rules in the same subset as well as their pseudo rules are then stored in an independent TCAM space. All subsets must be accessed to ensure that all the matching rules are extracted. The ruleset decomposition for minimizing the number of pseudo rules is an NP-hard problem. The authors approach this problem by applying Johnson's algorithm [9].

Some researchers achieve TCAM-based multi-match packet classification by modifying the hardware architecture. Faezipour and Nourani modify TCAM's priority encoder to produce all matching entries by employing some additional circuits [10]. They also propose a two-phase scheme, maximum-minimum intersection partitioning (MX-MN-IP), to divide a rule set, where the overlapping rules are stored in a partition. Each partition is further divided to smaller sub-partitions, where rules in the same sub-partition are disjoint to each other. All the matching rules of a search key can be generated by accessing all sub-partitions of a matching partition. Song and Lockwood proposed BV-TCAM [11], which integrates TCAM with BV [12] using FPGA. BV-TCAM removes priority encoder from TCAM to yield all the matching field specifications by storing the result of all match-lines in a bit vector. The matching rules can thus be determined by intersecting the bit vectors of all fields. FSBV avoids the slow range-matching procedures to optimize the combination of TCAM and SRAM for Snort rule sets [13]. It also relies on FPGA to implement the proprietary architecture. Shen et al. propose a hybrid scheme, which combines TCAM with SRAM in order

to alleviate storage requirement in TCAM [14]. The scheme stores distinct specifications of each field in a TCAM for indexing the match lists stored in SRAM. Because the number of distinct field specifications is usually much less than the number of rules, the scheme requires several small TCAM chips. The above proprietary architectures may increase implementation cost and complexity.

2.2 TCAM-Related Research

The research issues of the existing TCAM solutions include range encoding [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], power consumption [20], [25], [26], [27], [28], [29], [30], [31], storage reduction [32], [33], [34], [35], [36], and updates [27], [37], [38], [39]. The range encoding algorithms have two strategies, database-dependent and database-independent. Database-dependent encoding schemes have excellent efficiency for both TCAM entry length and count, but they require an extra mapping procedure to retrieve values of encoded range fields [7], [16], [17], [19], [40], [41]. Database-independent encoding schemes avoid extra TCAM searches, but range expansion may still occurs [5], [18]. Database-Independent Range Pre-Encoding (DIRPE) converts a range into a ternary string by using fence encoding, where a W-bit range field can be encoded into one (2^W-1)-bit ternary string. Because there could be no enough available bits in a TCAM entry for encoding a wide field, DIRPE divides a range field into multiple r-bit subfields, where each subfield is independently encoded. As a result, the length of the discriminator field is equal to $\lceil W/r \rceil (2^r - 1)$ with $\lceil W/r \rceil$ subfields. The tradeoff of a shorter discriminator field is the increased expansion factor, where a range is expanded to $2(\lceil W/r \rceil - 1)$ ternary strings in the worst case. The range expansion problem increases the cost of performing multi-match classification.

Because TCAM is designed for reporting the first matching entry, some low-priority rules are useless since they will not be reported for any packets. Some entry pruning algorithms remove these redundant rules to reduce the number of TCAM entries[42]. These algorithms cannot be applied to multi-match packet classification since all matching rules are required.

Minimizing the cost of updating TCAM entries is important for single-match packet classification. To ensure the correctness of packet forwarding, TCAM entries must be sorted according to their priorities in a descending order. A new rule must occupy a correct position to conform the predefined priority. In [37], two algorithms, PLO_OPT and CAO_OPT, are proposed to reduce update cost. CoPTUA ensures the correctness of rule matching without locking rule table by maintaining the consistency of rules [38]. PC-DUOS divides a rule set into two subsets according their priorities [27]. The rules of each subset are stored in different TCAMs to shorten update time. Multi-match packet classification may also have to deal with the problem of updates. For example, both GI and SSA generate pseudo rules. Inserting a new rule or deleting an existing rule may result in updating multiple TCAM entries.

3 Motivation

Before proceeding to our scheme, we present our motivation based on the observations to MUD and GI in terms of speed and storage performance.

As mentioned above, MUD is a space-efficient algorithm, which achieves multi-match packet classification by appending a discriminator to each TCAM entry. MUD requires multiple TCAM accesses to yield all the matching rules. The number of TCAM accesses is dominated by two factors, the number of matching rules and the cost of representing a discriminator. When an entry matches a search key, MUD assigns a range larger than the discriminator value of the matching entry to the discriminator field of the search key for yielding the next matching rule. The discriminator range in a search key must be transformed into a TCAM-compatible format. As a result, the range could be converted to multiple ternary strings where each string corresponds to one search key. Each search key requires one TCAM access but does not always yield a matching rule. The TCAM accesses can be reduced by employing efficient range encoding algorithm. Assume that the maximum number of matching rules is m. Then, MUD requires at least m unique discriminator values and a $\log_2 m$-bit discriminator field is required. In our experiments, we found that six-bit discriminator field is enough for the current rule sets. A six-bit range could be expanded to eleven prefixes in the worst case to result in excessive number of search keys as well as redundant TCAM accesses.

The excessive number of TCAM entries can be reduced by performing range encoding. Both types of encoding algorithms have their own costs for encoding the discriminator ranges. Database-dependent encoding schemes require m mappings, and database-independent encoding schemes may result in redundant TCAM accesses. In this paper, we attempt to eliminate the redundant TCAM accesses caused by database-independent encoding schemes.

Currently, a router supporting packet classification is usually equipped with TCAM chips. Although the number of rules that network administrators can insert is limited by the capacity of these chips, some routers may reserve TCAM entries for lowering update latency. Since multi-match packet classification generates all matching rules, rule reordering is unnecessary. These entries can thus be used to boost the speed performance of TMPC.

We note that for a TCAM chip, the percentage of occupied TCAM entries does not directly correlate to power consumption. The state-of-the-art TCAM chips use a valid bit to control the availability of each entry. Although the invalid TCAM entries can be kept away from being compared to lower power consumption, there is still significant power consumption for leakage power of SRAM cells, capacitance of searchline and I/O registers [43]. As a result, the invalid TCAM entries still incur high power consumption, and the number of TCAM accesses still dominates the power consumption of TMPC.

We are motivated to improve the speed performance of MUD by exploiting the unused TCAM entries. We notice that previous algorithms of multi-match packet classification generate rule intersections to achieve superior speed performance. While a pure geometric-intersection scheme may result in a significant increase of TCAM entries, we can control the number of rule intersections by only intersecting a part of rules. The remaining rules are still maintained by MUD. There are two advantages for using our scheme with MUD. First, the maximum number of matching TCAM entries could be reduced because each rule intersection

TABLE 1
An Example with 10 Rules on Two Fields

Rule	f_1	f_2	Rule	f_1	f_2
R_0	$01*$	$*$	R_5	$10*$	$*$
R_1	1001	$*$	R_6	$1*$	$*$
R_2	$001*$	$*$	R_7	1100	$*$
R_3	$0*$	$*$	R_8	$00*$	$*$
R_4	$101*$	$*$	R_9	$*$	$0*$

merges two or more overlapping rules. As a result, redundant TCAM accesses could be eliminated. Second, the number of unique discriminator values could be decreased to reduce the number of chunks. Both advantages could benefit MUD to achieve better speed performance. It is worth to note that, because inserting a moderate number of TCAM entries would reduce the number of TCAM accesses, power consumption could be improved simultaneously.

4 Selectively Generating Pseudo Rules

4.1 Algorithm

Our scheme exploits the reserved TCAM entries to store rule intersections. To effectively utilize the reserved TCAM entries, we need to control the number of pseudo rules. There are two possible approaches. The first approach generates pseudo rules for all rules first. Then, it gradually chooses the rules overlapping with the most rules. These rules are handled by MUD for reducing the pseudo rules. The procedure stops until both intersection and original rules can be stored in TCAM. This approach is suitable for the rule set whose pseudo rules are fewer or slightly more than the available TCAM entries because only few rules experience extra process. It is not suitable for the rule sets with numerous rule intersections. Because the number of rule intersections would grow exponentially with respect to the number of overlapping rules, the computation cost of this approach could be too high to be feasible. In addition, this approach cannot effectively determine the length of the discriminator field. If these chosen rules need numerous discriminator values, the speed performance of MUD is degraded, as mentioned above.

We choose another approach that stores the original rules using MUD and selectively generates pseudo rules for some rules. To reduce the number of discriminator values in MUD, the original rules with the same x-bit distributions are categorized in a group, where two rules have the same x-bit distribution if the occurrences of their x-bits are the same. We note that any two rules with the same x-bit distribution are either mutually disjoint or identical. Since the identical rules are always merged to avoid redundant TCAM entries, the rules in a group can share the same discriminator value. Accordingly, we iteratively choose two groups from MUD to store their rule intersections in TCAM. By controlling the number of the generated rule intersections, we can simultaneously reduce both numbers of rule groups and the required discriminator values to improve the speed performance of MUD. Subsequently, the detailed procedure of our approach is described.

We start from introducing the MUD procedure of rule grouping. As mentioned above, MUD appends a discriminator field to each TCAM entry so that a search key can be combined with different discriminator ranges to determine the TCAM entries for comparison. To generate the discriminator value of each rule, the range fields of the original rules are converted to ternary strings by using database-independent range pre-encoding. Then, the rules with the same x-bit distributions are categorized in a group. Two rules with the same x-bit distribution must be disjoint to each other unless they have the same specifications. Accordingly, a packet matches at most one rule in a group. The rules in the same group are assigned with the same discriminator value; hence, the length of the discriminator field is reduced to $\log_2(number\ of\ distinct\ x-bit\ distributions)$ bits.

TABLE 2
Groups of the Rules in Table 1

G0	G1	G2	G3	G4
$R_9:(*,0*)$	$R_2:(001*,*)$	$R_5:(10*,*)$	$R_3:(0*,*)$	$R_1:(1001,*)$
	$R_4:(101*,*)$	$R_8:(00*,*)$	$R_6:(1*,*)$	$R_7:(1100,*)$
		$R_0:(01*,*)$		

We use an example with ten rules in Table 1 to further explain the above procedure. Both fields of these rules have different prefix lengths. These rules are categorized to distinct groups according their x-bit distributions. The result of rule grouping is listed in Table 2. Rules in the same group have the same x-bit distribution. For example, R_5, R_8 and R_0 have the same x-bits in the least-significant two bits of f_1 and all bits of f_2. These rules are disjoint to each other.

Next, we use a greedy method to selectively store the rule groups in an intersection group. Since the rules from different groups could overlap with each other, pseudo rules are generated for the overlapping rules. All rules in the intersection group share the same discriminator value. The pseudo rules must be located in front of the original rules. Unlike the original groups of MUD, a packet could match more than one rule of the intersection group. A packet that matches more than two original rules always matches the pseudo rule, which is synthesized for the overlapping region of all the matching rules.

Due to the variant characteristics of different rule sets, a rule set may have variable number of groups and variable number of rules in each group. We also note that although the rules are categorized according to their x-bit distributions, the rules in different groups are not necessary to overlap with each other. It is possible to merge several groups in an intersection group with the cost of few pseudo rules. Our algorithm aims at merging as more groups as possible under the limited number of available TCAM entries.

We describe the procedure of selecting rule groups to be inserted into the intersection group. Initially, the intersection group is empty. We pick the smallest group and insert its rules into the intersection group. The smallest group stands for the number of the required TCAM entries for the rules in the group. If there are two or more groups having the same size, one of them is randomly selected. Next, another group is selected to be merged with the intersection group. The group should be the one that leads to the fewest extra TCAM entries. The calculation is performed by first generating all pairs of partially overlapping rules. For each pair, a pseudo rule is generated for the overlapping region. The number of the required TCAM entries for each pseudo rule is then produced. The group that generates the least

TABLE 3
Pseudo Rules Generated by the Rules of Each Different Group

G1	G2	G3	G4
$R_{2,9} : (001*, 0*)$	$R_{5,9} : (10*, 0*)$	$R_{3,9} : (0*, 0*)$	$R_{1,9} : (1001*, 0*)$
$R_{4,9} : (101*, 0*)$	$R_{8,9} : (00*, 0*)$	$R_{6,9} : (1*, 0*)$	$R_{7,9} : (1100*, 0*)$
	$R_{0,9} : (01*, 0*)$		

TABLE 4
The Final Result of Our Scheme

IG: 3	G2: 0	G3: 1	G4: 2
$R_{2,9} : (001*, 0*)$	$R_5 : (10*, *)$	$R_3 : (0*, *)$	$R_1 : (1001, *)$
$R_{4,9} : (101*, 0*)$	$R_8 : (00*, *)$	$R_6 : (1*, *)$	$R_7 : (1100, *)$
$R_9 : (*, 0*)$	$R_0 : (01*, *)$		
$R_2 : (001*, *)$			
$R_4 : (101*, *)$			

TCAM entries for the extra pseudo rules is merged into the intersection group. If more than one candidate group generates the fewest TCAM entries, then we choose one of them randomly. Because these TCAM entries will occupy the unused TCAM entries, we must check whether the number of the TCAM entries yielded from the pseudo rules is smaller than the number of available TCAM entries. If there are enough TCAM entries, the pseudo rules are generated and stored. Otherwise, our greedy method stops merging with additional groups and is complete.

We use the previous example in Table 2 to illustrate the procedure of group selection. There are five groups with different sizes as shown in Table 2. In the first iteration, the group G0 is the only candidate because it is the smallest group. We store its rules in the intersection group. In the second iteration, we calculate the cost of inserting different groups into the intersection group. We list the generated pseudo rules for moving the rules of different groups to the intersection group in Table 3, where the costs for groups G1 $\sim$ G4 are two, three, two and two. Because three groups, G1, G3 and G4, have the same cost, our greedy method randomly selects the rules of G1 to be stored in the intersection group. Assume that there is not enough TCAM entries for merging another group, the above procedure stops. A total of one intersection group and three original groups, as shown in Table 4, result from implementing our algorithm.

Each group is then assigned with a unique discriminator value, and the intersection group is always assigned with the largest discriminator value. Since we merge multiple rule groups in an intersection group, the length of the discriminator value can be shortened. In the previous example, each discriminator value is reduced from three bits for five groups to two bits (see Table 4). The rules are sorted according to the discriminator values of their groups and inserted into TCAM by appending the discriminator values of their groups.

Because the intersection group occupies the empty space of TCAM, its rules are stored in the bottommost TCAM entries (i.e., behind the empty space). Moreover, the pseudo rules are stored in front of the original rules to ensure the correctness of the search procedure. As mentioned above, the rules of the intersection group may overlap with each other; thus, a search key may match more than one TCAM entry. By assigning the same discriminator value to the TCAM entries of the intersection group, only the first matching entry is accessed and the other matching entries are ignored.

Rule updates taking place at the original groups or the intersection group can be efficiently processed. Any rule insertion or deletion in the original groups can be accomplished by updating at most one TCAM entry for each group [37]. The intersection group can also be updated by inserting new pseudo rules for rule insertions or deleting the existing pseudo rules for rule removals. If a new rule generates too many pseudo rules, the new rule should be stored in a new rule group behind the original rule groups to avoid high update cost.

4.2 Implementation Considerations

In our scheme, both a discriminator field and an original rule are stored in one TCAM entry. In a typical TCAM, each entry has 144 bits. A five-field (source/destination IP address, source/destination port number, protocol) rule consumes 104 bits. Thus, there are 40 available bits. These bits can be used to improve the efficiency of range encoding. We assume that the range fields (source and destination port fields) of all rules are encoded by using the DIRPE algorithm. DIRPE can reduce the number of TCAM entries for both original rules and pseudo rules. Since each field of a pseudo rule is equal to one of the field specifications of two overlapping rules, a succinct representation of range fields can thus reduce the number of required TCAM entries for the pseudo rules. For both port fields, source-port field uses six subfields whose strides are 2, 2, 3, 3, 3, 3 and destination-port field uses seven subfields whose strides are 2, 2, 2, 2, 2, 3, 3. Hence, both fields consume 34 bits and 29 bits, respectively. After the deduction of 32 bits from both ports, DIRPE requires 31 extra bits.

There are nine bits left for the discriminator field. As mentioned above, the six-bit discriminator field, which can identify at most 64 groups, is split into three two-bit subfields, where each subfield requires three bits for DIRPE. If there are more than 64 groups, the number of bits used both port fields can be reduced by increasing the number of subfields for DIRPE. For example, the source-port field can be split into seven subfields as the destination-port field to produce five extra bits. With these extra bits, the discriminator field can specify up to 1,024 groups.

4.3 Search Procedure

To perform multi-match packet classification upon the TCAM entries, the operations to generate the search key are listed below. First, all the inspected header fields of an incoming packet are extracted and concatenated as a search key. Then, both transport-layer port fields of the search key are encoded by using DIRPE. Next, a discriminator field is appended to the search key. Initially, all bits in the discriminator field are set to 'x' for comparing all TCAM entries.

In the first TCAM access, all TCAM entries are compared. If a matching entry is reported, the discriminator value of the matching entry is retrieved. Assume that the value is j. Then, the discriminator value of the search key is revised to "greater than j" in order to search for the next

TABLE 5
The Rules with Overlapping Pre-Computation

IG: 3	G2: 0	G3: 1	G4: 2
$R_{2,9}:(001*,0*)/\phi$	$R_5:(10*,*)/1$	$R_3:(0*,*)/3$	$R_1:(1001,*)/3$
$R_{4,9}:(101*,0*)/\phi$	$R_8:(00*,*)/1$	$R_6:(1*,*)/2$	$R_7:(1100,*)/3$
$R_9:(*,0*)/\phi$	$R_0:(01*,*)/1$		
$R_2:(001*,*)/\phi$			
$R_4:(101*,*)/\phi$			

TABLE 6
The Rule Groups with Discriminator Reassignment

IG: 51	G2: 15	G3: 31	G4: 47
$R_{2,9}:(001*,0*)$	$R_5:(10*,*)$	$R_3:(0*,*)$	$R_1:(1001,*)$
$R_{4,9}:(101*,0*)$	$R_8:(00*,*)$	$R_6:(1*,*)$	$R_7:(1100,*)$
$R_9:(*,0*)$	$R_0:(01*,*)$		
$R_2:(001*,*)$			
$R_4:(101*,*)$			

matching entry within the remaining TCAM entries. The discriminator range "greater than j" is encoded by using DIRPE to reduce the number of TCAM accesses. The maximum number of the subranges for "greater than j" is equal to the number of chunks used in DIRPE. For example, we use three two-bit chunks for encoding the discriminator field. If there is a matching entry with a discriminator value, 1, then the discriminator ranges encoded by DIRPE are "000 000 x11", "000 xx1 xxx" and "xx1 xxx xxx", where the corresponding decimal ranges are [2:3], [4:15] and [16:63]. One TCAM access is performed for each range. The above procedure repeats until that no match is reported.

4.4 Improving Speed Performance

We develop two speed refinements for MUD. The first refinement, overlapping pre-computation, is based on the observation to the difference between the number of matching rules and the number of TCAM accesses. The main reason for the difference is the range representation of the discriminator field. While the discriminator uses the index number of a matching rule to determine the range of TCAM entries for further comparisons, we could bypass several rule groups by using pre-computation information. For each rule, we calculate the smallest index number of the subsequent groups which have at least one overlapping rule. This number denotes the next group with possible matching rules. We can store the smallest index number of the overlapping groups in the corresponding entry of SRAM. When a rule matches the search key, the corresponding index number is extracted to revise the range of compared TCAM entries. Since each TCAM access must retrieve the TCAM entries whose discriminator values are equal to or larger than the index number, those groups without any overlapping rules are ignored. This approach can reduce the number of TCAM accesses but does not yield any matching rules. For the example in Table 5, with the pre-computation information behind each slash, R_3 does not overlap with any rules in G4 but overlaps with more than one rule in the intersection group whose index value is 3. G4 can thus be bypassed if R_3 is matched by the search procedure.

The second approach, discriminator reassignment, is designed for the proposed scheme to reduce the number of DIRPE entries for a given range. Since the proposed scheme reduces the number of groups to be searched by using MUD, we develop a refinement for further speed improvement. By observing the multi-bit trie for multiple chunks of fence encoding, there are several ranges, which can be represented by one ternary string. In Section 4.2, we use DIRPE with three two-bit chunks for the discriminator field. With the DIRPE configuration, there are nine ranges whose DIRPE codes only need one entry. These ranges in decimal numbers are [16:63], [32:63], [48:63], [52:63], [56:63], [60:63], [61:63], [62:63], and [63:63]. The largest values in front of these ranges can be used to label the groups of MUD to reduce TCAM accesses. For example, we can assign a group with an identifier value, 15. When there is a matching rule in the group, the search procedure can use the DIRPE code of [16:63], namely "xx1 xxx xxx", to express the range for the subsequent search. In this example, there is a total of ten values for labeling groups, including the last number, 63. When the rules of the groups labeled based on the perfect DIRPE codes are matched, the range corresponding to the subsequent groups can be encoded as one DIRPE code. As a result, only one TCAM access is required for each discriminator range. In our example, the number of remaining groups is smaller than ten, and each group can be assigned a new group ID, as shown in Table 6. With these group IDs, only one TCAM access is needed to search for the remaining groups. If there remain more than ten groups, the extra groups could be labeled as those values whose successive ranges can be encoded as two DIRPE codes.

TABLE 7
The Statistics of SNORT Rulesets

SNORT	Rules	DIRPE	Groups	Avg. Acc.	Max. Acc.
v2.9.0.0	956	1,207	22	10.031	16
v2.9.1.0	1,128	1,417	24	13.585	21
v2.9.2.0	1,172	1,469	25	12.299	23
v2.9.3.0	1,222	1,536	27	11.564	24

5 Performance Evaluation

In this section, we evaluate the performance of our scheme, selective generation of pseudo rules (SGPR), with different types of rulesets. We start our evaluation by using four SNORT rulesets [23] whose versions vary from v2.9.0.0 to v2.9.3.0. The initial statistics of four Snort rule sets are shown in Table 7. There are about 1,000 to 1,200 rules in these Snort rulesets. By encoding the range fields of the original rules with DIRPE, these rulesets consume about 1,200 to 1,500 TCAM entries. These rules are categorized into $22 \sim 27$ groups. We also show the speed performance of MUD by listing the number of memory accesses in both average and worst cases. The maximum TCAM accesses of MUD is proportional to the number of rule groups.

SGPR uses free TCAM entries to improve the performance of TMPC; thus, our first experiment demonstrates the relationship between the performance of SGPR and the number of free TCAM entries. Because each rulesets have different sizes, we use a storage expansion ratio to determine the number of available TCAM entries for storing a ruleset. The expansion ratios are 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 4 and 8.

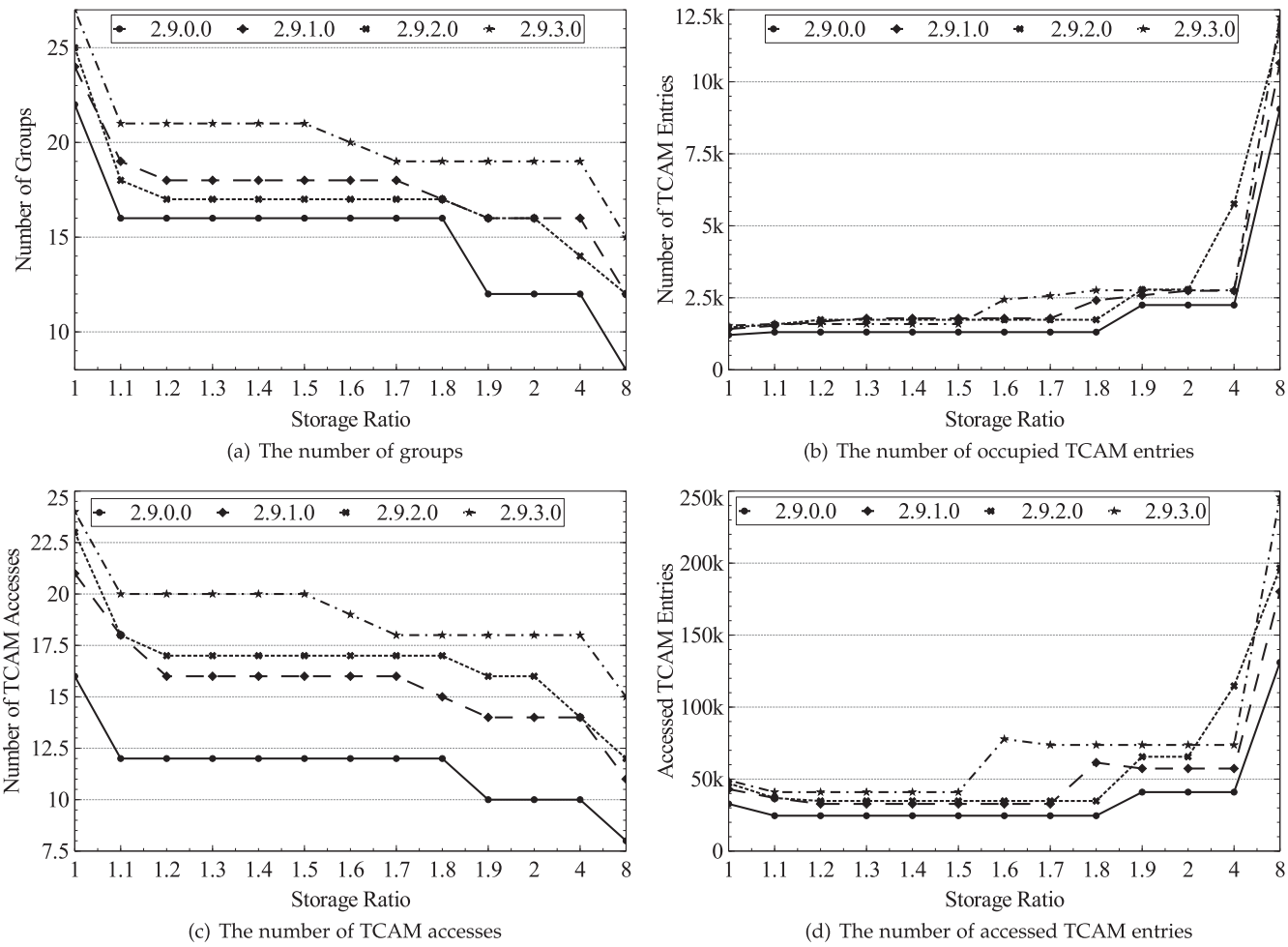

Fig. 1. Performance of SGPR with different expansion ratios.

The number of TCAM entries used by SGPR is thus equal to the number of rules with DIRPE times the expansion ratio. We evaluate the tradeoff between space and speed performance of SGPR by adjusting the number of occupied TCAM entries. Accordingly, we evaluate four performance metrics related to SGPR in Fig. 1. These performance metrics include the numbers of rule groups, total TCAM entries, TCAM accesses per classification and accessed TCAM entries per classification.

We first show the number of rule groups generated by SGPR in Fig. 1a. When the expansion ratio is equal to one, SGPR has the same number of groups as MUD because no groups can be merged. The number of groups for all rulesets can be significantly reduced with 10 percent extra TCAM entries by setting the expansion ratio to 1.1. However, the reduction of rule groups is not effective when the expansion ratio is increased to 1.2, where only one group of two rulesets, v2.9.1.0 and v2.9.2.0, can be merged. For the ruleset, v2.9.3.0, another group reduction takes place until the expansion ratio is increased to 1.6, which means the group reduction requires about 40∼50% extra TCAM entries. We conclude that the effectiveness of group reduction is not the same for all rulesets because of different overlapping relationships between rule groups.

When the expansion ratio is increased, SGPR may consume more TCAM entries to merge rule groups, as shown in Fig. 1b. We note that two groups would not be merged if the merge requires more TCAM entries than available. However, it is possible that the TCAM entries of SGPR could be the same for different expansion ratios. For example, the number of TCAM entries for all rulesets remains the same for the expansion ratio within 1.2 to 1.5 because no any two groups can be merged. The increment of TCAM entries is significant when the expansion ratio is large enough.

The speed performance of SGPR is tied to the number of rule groups. In the worst case, the search procedure accesses all groups to yield all matching rules. As a result, the number of TCAM accesses is tied to the number of rule groups, as shown in Fig. 1c. By increasing the expansion ratio, the speed performance of SGPR can be improved. Similarly, the expansion ratio 1.2 improves speed performance for all rulesets.

In Fig. 1d, we show the number of accessed TCAM entries per classification in the worst case. We assume that the smallest TCAM chip which can store the required TCAM entries is used to store the entries. The number of TCAM entries per classification is yielded by multiplying the number of TCAM accesses to the number of entries in the TCAM chip because the invalid TCAM entries still consume a considerable amount of power, as mentioned in Section 3. The smallest TCAM chip has 1,024 entries. If a ruleset has more than 1,024 entries, the TCAM chip could be exponentially enlarged by the power of two. Thus, the second smallest TCAM has 2,048 entries and so on. Because different rulesets have a different number of overlapping rules between two rule groups, they

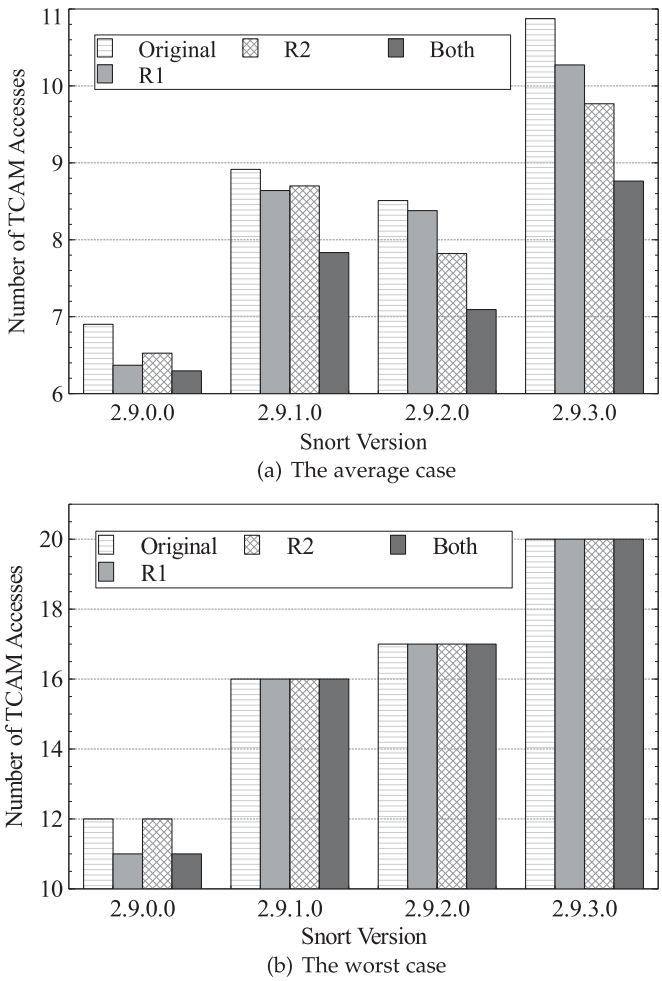

Fig. 2. Speed performance of SGPR with different refinements.

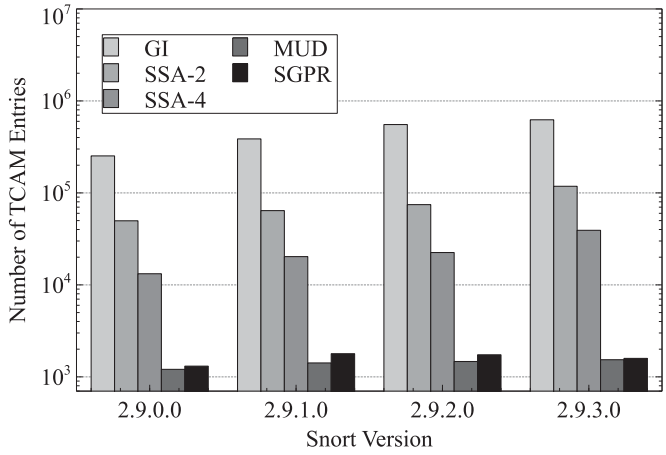

Fig. 3. The number of TCAM entries for different schemes.

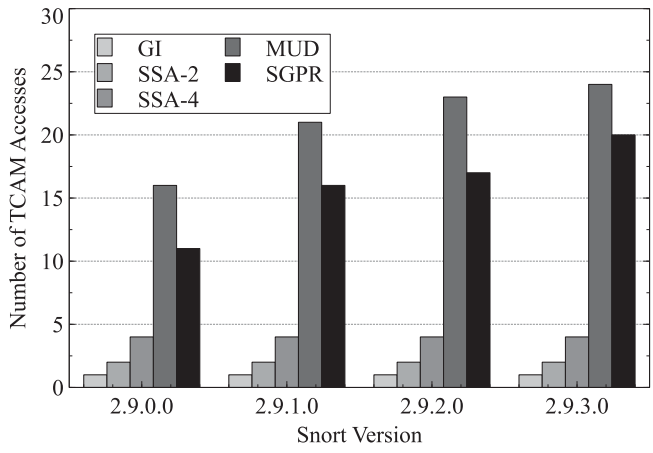

Fig. 4. The number of TCAM accesses in the worst case for different schemes.

may need TCAM chips with different sizes even with the same expansion ratio. For the rulesets requiring a larger TCAM chip, the number of accessed TCAM entries would increase sharply. For example, the number of accessed entries for Snort v2.9.3.0 for the expansion ratio 1.6 is almost twice that for the expansion ratio 1.5. Although the number of TCAM accessed is reduced, the adoption of a larger TCAM chip may result in a significant increment of accessed TCAM entries. A small expansion ratio is thus preferred to leverage the speed performance and power consumption simultaneously. Accordingly, we set the expansion ratio to 1.2 for SGPR in the following experiments.

Next, we show the performance improvement achieved by two refinements presented in Section 4.4. We denote the first refinement, overlapping pre-computation, as R1 and the other refinement, discriminator reassignment, as R2. In Fig. 2, four combinations, with or without applying each refinement, are considered. We also show the search performance in both average and worst cases. In Fig. 2a, both refinements can improve the speed performance of the original search procedure in the average case. R1 outperforms R2 for v2.9.0.0 and v2.9.1.0 but R2 is better for the other two rulesets. Because the performance of R2 is tied to the number of rule groups generated by SGPR, the rulesets with many groups, e.g., v2.9.3.0, cannot be fully benefited from the reassigned discriminator values. Implementing both refinements always achieves the best performance among these four combinations.

Although the proposed refinements can reduce the number of TCAM accesses in average case, they are not equivalently effective in the worst case. As shown in Fig. 2b, R1 can only reduce the number of TCAM accesses for Snort v2.9.0.0, and R2 does not demonstrate any improvement. The worst-case performance occurs when there is one matching rule in each group yielded by SGPR. It is because the rule groups of SGPR usually heavily overlap with each other. Since both refinements attempt to eliminate TCAM accesses without any matching rule, they cannot reduce the number of TCAM accesses in the worst case.

We relate the performance of SGPR with three previous schemes, GI [4], SSA [8] and MUD [5]. Because SSA can reduce pseudo rules by generating more rule subsets, we include two configurations for SSA, two subsets (SSA-2) and four subsets (SSA-4), to show the tradeoff between storage and speed performance. Three performance metrics, the numbers of total TCAM entries, TCAM accesses per classification and accessed TCAM entries per classification, are revealed in Figs. 3, 4, and 5.

First, we discuss the storage performance of different schemes in Fig. 3. Because there are a lot of overlapping rules in Snort rulesets, GI generates a great number of pseudo rules. By splitting the overlapping rules in a ruleset into different

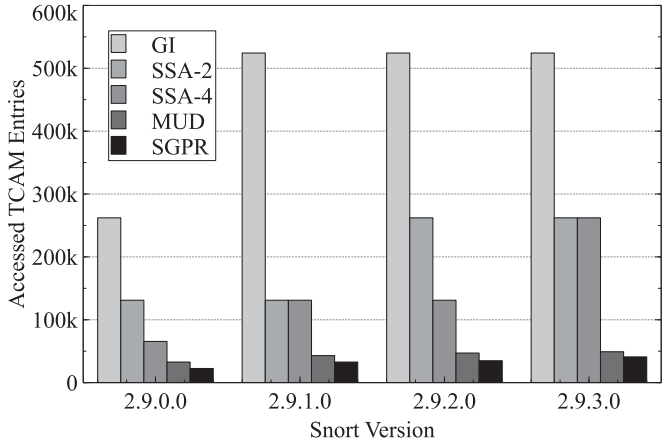

Fig. 5. The number of accessed TCAM entries per classification in the worst case for different schemes.

TABLE 8
The Statistics of ClassBench Rulesets

Database		Rules	DIRPE	Avg. Acc.	Max. Acc.
ACL	REAL	683	1,168	8.465	37
	1K	916	1,112	3.948	7
	5K	4,415	5,516	7.537	13
	10K	9,603	11,746	6.153	10
FW	REAL	269	443	3.547	4
	1K	791	1,413	3.915	8
	5K	4,653	7,351	4.173	8
	10K	9,311	14,847	4.841	6
IPC	REAL	1,550	1,791	6.726	16
	1K	938	1,047	4.943	11
	5K	4,460	5,008	6.152	14
	10K	9,037	10,203	5.916	14

subsets, SSA can reduce the number of pseudo rules to achieve better storage performance. Without generating any pseudo rules, MUD requires the fewest TCAM entries among the four schemes. SGPR has better storage performance than GI and SSA for the Snort rulesets because it produces a moderate number of pseudo rules in a controlled manner.

We show the number of TCAM accesses in the worst case in Fig. 4. There exists a significant tradeoff between speed and storage performance in the previous schemes. The speed performance of the previous schemes shows a reverse trend as the storage performance, where GI has the best performance and MUD requires the most TCAM accesses. The speed performance of SSA-2 and SSA-4 is equal to the number of rule subsets. SGPR improves the speed performance of MUD by reducing the number of rule groups. In particular, SGPR reduces $16\% \sim 30\%$ TCAM accesses required by MUD. As mentioned above, the speed performance of SGPR is tied to the number of generated rule groups. Due to the heavily overlapping rules in Snort rulesets, the limited number of pseudo rules also restricts the speed improvement. We note that SGPR can be treated as a generalized version of GI, where GI merges all rule groups in one. If there are enough TCAM entries, SGPR can achieve the same speed performance as GI.

Fig. 5 depicts the number of accessed TCAM entries in the worst case for different schemes. Although the results are yielded by multiplying the number of TCAM entries in a chip by the number of TCAM accesses, storage performance dominates the power consumption of TMPC. Although GI only needs one TCAM access to produce the matching rules, it consumes too many TCAM entries to result in the highest power consumption among four schemes. SSA-2 can reduce 50 percent or more accessed TCAM entries of GI because SSA needs one fourth or less TCAM entries of GI. However, SSA-4 does not improve power consumption of SSA-2 for Snort v2.9.1.0 and v2.9.3.0 because the storage reduction cannot compensate for the extra TCAM accesses. MUD outperforms the previous schemes because of its superior storage performance. SGPR further reduces the number of TCAM accesses for MUD by generating a moderate number of pseudo rules. While SGPR does not incur more cost than MUD, it provides a feasible approach to improve both speed performance and power consumption.

We further evaluate the performance of our scheme based on three types of rule databases used in [44]. These types include access control list (ACL), firewall (FW) and IP chain (IPC). There is one real database for each type. Because these real databases are too small to evaluate the scalability, another three databases, which have 1,000, 5,000 and 10,000 rules, are synthesized by using ClassBench [45]. Some rules of these databases are removed because of duplication. Due to the different characteristics and configurations, each rule database consumes different number of TCAM entries. In particular, the number of TCAM entries with DIRPE varies from about 450 to 15,000. We list the statistics of different rule databases in Table 8, where the speed performance of MUD is also included.

We first show the storage performance of SGPR in Fig. 6, where the numbers of required TCAM entries for GI, SSA and MUD are also revealed. Among all schemes, MUD has the best storage performance and GI has the worst, but their differences vary for different database types. For example, GI generates few pseudo rules for the ACL databases because there are fractional overlapping rules in these databases. The IPC databases have more overlapping rules than the ACL databases to result in more TCAM entries for GI. While the FW databases have the most overlapping rules among all databases, the rules of these databases also specify wide port ranges (e.q., 1024-65535) to result in the worst storage performance for GI. The difference between GI and MUD also depends on the number of rules. For example, GI requires about fifty times more TCAM entries than MUD does for the 1K-rule FW database, but the difference is increased to more than one thousand for the 10K-rule FW database. As a result, GI could not be able to support large rule databases due to limited TCAM storage. Although SSA can significantly improve the storage performance of GI, it still requires near one hundred times more TCAM entries than MUD does for the 10K-rule FW database. The largest TCAM chip with 72 Mb cannot accommodate the entries required by 5K-rule and 10K-rule FW databases. As compared to GI and SSA, SGPR generates a moderate number of pseudo rules by setting the rule expansion ratio to 1.2. Therefore, SGPR can keep the storage performance at the same level as MUD.

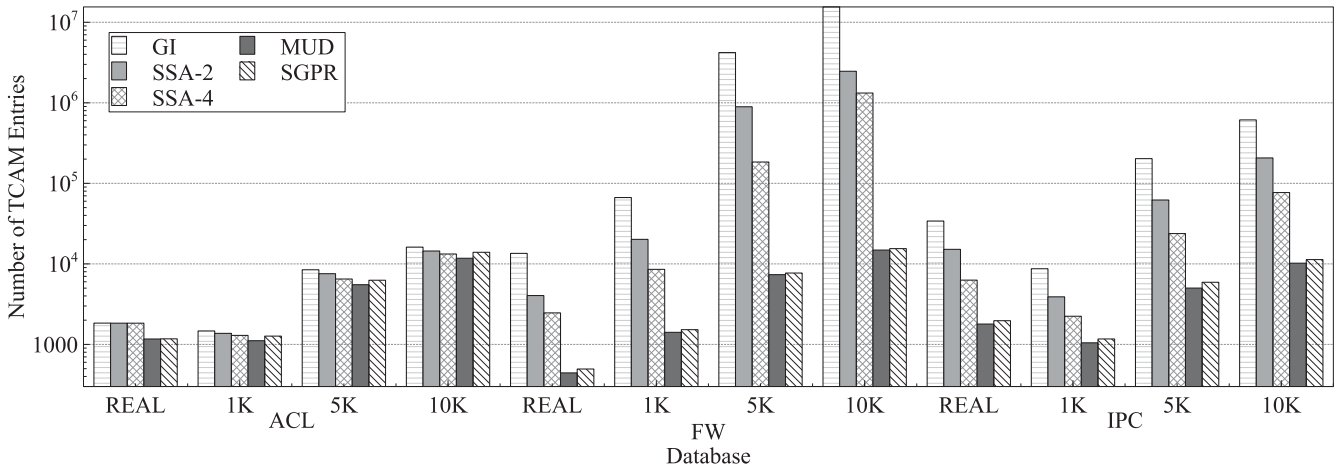

Fig. 6. The number of TCAM entries for different schemes.

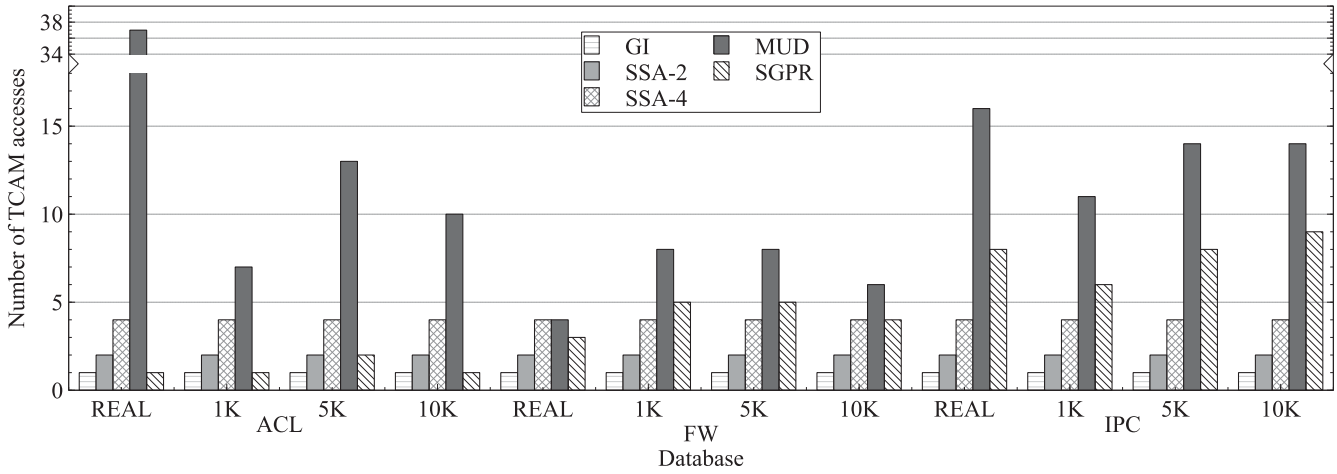

Fig. 7. The number of TCAM entries accesses in the worst case for different schemes.

In Fig. 7, we show the number of TCAM accesses in the worst case for different schemes. Similar to what is observed in Snort rulesets, GI has the best speed performance and MUD has the worst. For the ACL databases, GI has the best overall performance because only few pseudo rules are required. However, its speed performance could not be achieved for FW databases due to the high storage overhead. SSA can improve the feasibility by balancing the storage and speed performance. The speed performance of MUD is tied to the number of rule groups; thus, MUD has good speed performance for the FW databases. For the ACL and IPC databases, MUD needs twofold or more TCAM accesses than SSA. SGPR can effectively leverage the available TCAM entries to lower the number of rule groups. SGPR requires at most five TCAM accesses for the ACL and FW databases and less than ten accesses for the IPC databases. For the ACL databases, SGPR achieves similar speed performance as GI and outperforms SSA. SGPR has worse speed performance than SSA for the FW and IPC databases as the tradeoff for lower storage requirements. As compared to MUD, SGPR drastically improves its speed performance by selectively merging rule groups.

Finally, we show the number of accessed TCAM per classification in the worst case for different schemes in Fig. 8. Unlike the experimental results of Snort rulesets, the power consumption for the ACL databases are dominated by speed performance. Because there are fewer overlapping rules in the ACL databases, GI and SGPR have the best performance, and MUD accesses the most TCAM entries. The performance of GI and SSA severely degrades for the FW databases due to the overwhelming number of overlapping rules. In addition, SSA cannot outperform GI even with a slower speed performance. MUD and SGPR achieve better power consumption than GI and SSA for the FW databases because of significantly better storage performance. SGPR outperforms MUD due to the smaller number of required TCAM accesses. Similar results can also be observed for 5K-rule and 10K-rule IPC databases.

In summary, SGPR provides superior storage performance for all types of databases. It also achieves comparable speed performance as GI and SSA for the databases with few overlapping rules. For the databases with more overlapping rules, the scalability of SGPR is better than GI and SSA because SGPR can adjust the number of pseudo rules according to the number of available TCAM entries. Both GI and SSA cannot support heavily overlapping rules because of their huge memory footprints. As compared to MUD, SGPR always has better speed performance by selectively merging rule groups, especially for the rule databases

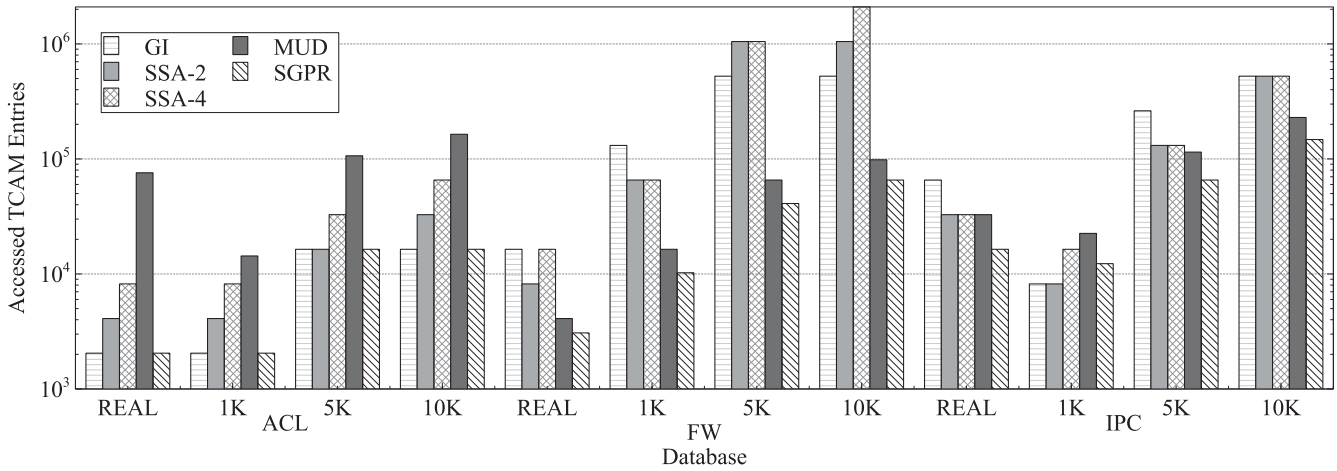

Fig. 8. The accessed TCAM entries per classification in the worst case for different schemes.

with few overlapping rules. We also conclude that SGPR has the best power consumption among these schemes for databases heavily embedded with overlapping rules.

6 Conclusions

The existing TCAM-based algorithms for multi-match packet classification have significant tradeoff between speed and space performance. They may result in high memory requirements or require multiple TCAM accesses to find out all the matching rules. Both drawbacks not only affect the feasibility and throughput but also increase the power consumption of packet classification. In this paper, we present a scheme to achieve efficient TCAM-based mulit-match packet classification by selectively generating pseudo rules. Our scheme is based on the observation that a fixed-size TCAM chip usually has unused entries after storing the original rules. Since these entries still consume considerable amount of energy, it would be reasonable to use these entries to improve the speed performance of packet classification. Accordingly, our algorithm generates pseudo rules to merge rule groups. As a result, the number of matching TCAM entries per classification can be reduced to minimize the number of required TCAM accesses. Moreover, because fewer number of unique discriminator values are required, the number of chunks for the discriminator field with DIRPE can be reduced to avoid redundant TCAM accesses. Our experimental results show that, as compared to MUD, our scheme has similar memory requirements but much better speed performance. Our scheme also needs fewer TCAM entries than the existing intersection-based schemes. In summary, our scheme has superior scalability with respect to the number of overlapping rules for the performance metrics of interests.

Acknowledgments

The authors would like to thank Prof. An-Yeu (Andy) Wu, Dr. Ding-Yuan Lee, and Dr. Ting-Sheng Chen at the Access IC Design Laboratory in National Taiwan University for their valuable information about TCAM power consumption. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Grant MOST 105-2221-E-005-046 and 106-2221-E-005-015.

References

[1] E. Norige, A. X. Liu, E. Torng, E. Torng, E. Norige, and A. X. Liu, "A ternary unification framework for optimizing TCAM-based packet classification systems," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 657–670, Apr. 2018.

[2] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Survey*, vol. 37, no. 3, pp. 238–275, 2005.

[3] B. Agrawal and T. Sherwood, "Ternary CAM power and delay model: Extensions and uses," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 5, pp. 554–564, May 2008.

[4] Y. Fang and R. H. Katz, "Efficient multi-match packet classification with TCAM," in *Proc. Annu. IEEE Symp. High Perform. Interconnects*, 2004, pp. 28–34.

[5] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 193–204, 2005.

[6] D.-Y. Chang and P.-C. Wang, "TCAM-based multi-match packet classification using multidimensional rule layering," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 1125–1138, Apr. 2016.

[7] J. V. Lunteren and T. Engbersen, "Fast and scalable packet classification," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 560–571, May 2003.

[8] F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz, "SSA: A power and memory efficient scheme to multi-match packet classification," in *Proc. ACM Symp. Archit. Netw. Commun. Syst.*, 2005, pp. 105–113.

[9] D. S. Johnson, "Approximation algorithms for combinatorial problems," *J. Comput. Syst. Sci.*, vol. 9, no. 3, pp. 256–278, 1974.

[10] M. Faezipour and M. Nourani, "Wire-speed TCAM-based architectures for multimatch packet classification," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 5–17, Jan. 2009.

[11] H. Song and J. W. Lockwood, "Efficient packet classification for network intrusion detection using FPGA," in *Proc. ACM/SIGDA 13th Int. Symp. Field-Programmable Gate Arrays*, 2005, pp. 238–245.

[12] T. Lakshman and D. Stidialis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 1998, pp. 203–214.

[13] W. Jiang and V. K. Prasanna, "Field-split parallel architecture for high performance multi-match packet classification using FPGAs," in *Proc. 21st Annu. Symp. Parallelism Algorithms Archit.*, 2009, pp. 188–196.

[14] R. Shen, X. Li, and H. Li, "A hybrid TCAM + SRAM scheme for multi-match packet classification," in *Proc. 13th Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, Dec. 2012, pp. 685–690.

[15] C. R. Meiners, J. Patel, E. Norige, E. Torng, and A. X. Liu, "Fast regular expression matching using small TCAMs for network intrusion detection and prevention systems," in *Proc. USENIX Conf. Secur.*, 2010, pp. 8–8.

[16] A. Bremler-Barr, D. Hay, and D. Hendler, "Layered interval codes for TCAM-based classification," in *Proc. IEEE INFOCOM*, 2009, pp. 1305–1313.

[17] Y.-K. Chang, C.-I. Lee, and C.-C. Su, "Multi-field range encoding for packet classification in TCAM," in *Proc. IEEE INFOCOM*, 2011, pp. 196–200.
[18] A. Bremler-Barr and D. Hendler, "Space-efficient TCAM-based classification using gray coding," in *Proc. IEEE INFOCOM*, 2007, pp. 1388–1396.
[19] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES: Dynamic range encoding scheme for TCAM coprocessors," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 902–915, Jul. 2008.
[20] A. Kesselman, K. Kogan, S. Nemzer, and M. Segal, "Space and speed tradeoffs in TCAM hierarchical packet classification," in *Proc. IEEE Sarnoff Symp.*, 2008, pp. 1–6.
[21] X. He, J. Peddersen, and S. Parameswaran, "LOP_RE: Range encoding for low power packet classification," in *Proc. IEEE 34th Conf. Local Comput. Netw.*, 2009, pp. 137–144.
[22] O. Rottenstreich and I. Keslassy, "Worst-case TCAM rule expansion," in *Proc. IEEE INFOCOM*, 2010, pp. 456–460.
[23] M. Roesch, "Snort - Lightweight intrusion detection for networks," in *Proc. 13th USENIX Conf. Syst. Admin.*, 1999, pp. 229–238.
[24] Z. Kai, H. Che, W. Zhijun, L. Bin, and Z. Xin, "DPPC-RE: TCAM-based distributed parallel packet classification with range encoding," *IEEE Trans. Comput.*, vol. 55, no. 8, pp. 947–961, Aug. 2006.
[25] T. Banerjee-Mishra and S. Sahni, "DUOS - Simple dual TCAM architecture for routing tables with incremental update," in *Proc. IEEE Symp. Comput. Commun.*, 2010, pp. 503–508.
[26] W. Lu and S. Sahni, "Low-power TCAMs for very large forwarding tables," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 948–959, Jun. 2010.
[27] T. Banerjee-Mishra, S. Sahni, and G. Seetharaman, "PC-DUOS: Fast TCAM lookup and update for packet classifiers," in *Proc. IEEE Symp. Comput. Commun.*, 2011, pp. 265–270.
[28] T. Banerjee-Mishra and S. Sahni, "PETCAM—A power efficient TCAM architecture for forwarding tables," *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 3–17, Jan. 2012.
[29] T. Banerjee-Mishra, S. Sahni, and G. Seetharaman, "PC-TRIO: An indexed TCAM architecture for packet classifiers," in *Proc. IEEE Symp. Comput. Commun.*, 2012, pp. 325–330.
[30] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," in *Proc. 11th IEEE Int. Conf. Netw. Protocols*, Nov. 2003, pp. 120–131.
[31] Y. Ma and S. Banerjee, "A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 335–346, Aug. 2012.
[32] C. R. Meiners, A. X. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2009, pp. 93–102.
[33] R. Wei, X. Yang, and H. J. Chao, "Block permutations in boolean space to minimize TCAM for packet classification," in *Proc. IEEE INFOCOM*, 2012, pp. 2561–2565.
[34] R. Cohen and D. Raz, "Simple efficient TCAM based range classification," in *Proc. IEEE INFOCOM*, 2010, pp. 1–5.
[35] C. R. Meiners, A. X. Liu, E. Torng, and J. Patel, "Split: Optimizing space, power, and throughput for TCAM-based classification," in *Proc. ACM/IEEE 7th Symp. Archit. Netw. Commun. Syst.*, 2011, pp. 200–210.
[36] A. Liu, C. Meiners, and E. Torng, "TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 490–500, Apr. 2010.
[37] D. Shah and P. Gupta, "Fast updating algorithms for TCAMs," *IEEE Micro*, vol. 21, no. 1, pp. 36–47, Jan. 2001.
[38] Z. Wang, H. Che, M. Kumar, and S. K. Das, "CoPTUA: Consistent policy table update algorithm for TCAM without locking," *IEEE Trans. Comput.*, vol. 53, no. 12, pp. 1602–1614, Dec. 2004.
[39] H. Song and J. S. Turner, "Fast filter updates for packet classification using TCAM," in *Proc. IEEE Global Commun. Conf.*, 2006, pp. 1–6.
[40] H. Liu, "Efficient mapping of range classifier into ternary-CAM," in *Proc. 10th Symp. High Perform. Interconnects*, 2002, pp. 95–100.
[41] Y.-K. Chang, C.-C. Su, Y.-C. Lin, and S.-Y. Hsieh, "Efficient gray-code-based range encoding schemes for packet classification in TCAM," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1201–1214, Aug. 2013.
[42] A. X. Liu, C. R. Meiners, and Y. Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *Proc. IEEE INFOCOM*, 2008, pp. 111–115.
[43] K. Nii, T. Amano, N. Watanabe, M. Yamawaki, K. Yoshinaga, M. Wada, and I. Hayashi, "A 28nm 400MHz 4-parallel 1.6Gsearch/s 80Mb ternary CAM," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2014, pp. 240–241.
[44] H. Song and J. S. Turner, "Toward advocacy-free evaluation of packet classification algorithms," *IEEE Trans. Comput.*, vol. 60, no. 5, pp. 723–733, May 2011.
[45] D. E. Taylor and J. S. Turner, "ClassBench: A packet classification benchmark," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 2068–2079.

Hsin-Tsung Lin received the MS degree in computer science and engineering from National Chung Hsing University, in 2012. He is currently working toward the PhD degree at the same university. His research interests include packet classification algorithms and software-defined networking.

Pi-Chung Wang received the MS and PhD degrees in computer science and information engineering from National Chiao Tung University, Taiwan, in 1997 and 2001, respectively. From 2002 to 2006, he was with Telecommunication Laboratories, Chunghwa Telecom. He joined the Deparement of Computer Science and Engineering (CSE), National Chung Hsing University (NCHU), Taiwan, in 2006. Since 2014, he has been a professor of CSE, NCHU. His research interests include IP lookup and classification algorithms, scheduling algorithms, protocols in local area networks, and wireless sensor networks. He is a member of the IEEE.