TCAM-Based Classification Using Divide-and-Conquer for Range Expansion

Hai Sun
Washington State University
Email: hsun1@eecs.wsu.edu

Yan Sun
Huawei US Research Center
Email: yan.yansun@huawei.com

Victor C. Valgenti
Petabi, Inc.
Email: vvalgenti@petabi.com

Min Sik Kim
Petabi, Inc.
Email: msk@petabi.com

***Abstract*—Ternary Content-Addressable Memory (TCAM) is the de facto industrial standard to perform packet classification. However inefficient representation of port ranges results in the range expansion problem which sharply degrades TCAM storage performance. A range has to be converted into a set of prefixes with each stored in a separate TCAM entry. The range expansion problem occurs when a rule with multiple range fields causes a multiplicative expansion in the number of TCAM entries. Unfortunately, the problem is growing worse as an increasing number of such rules in "real-world" classifiers are in use.**

To address range expansion our Divide-and-Conquer Scheme (DCS) fulfills the Divide-and-Conquer principle in two levels. First, we divide an individual range through range partitioning. A class of ranges can be optimally represented through a novel range encoding we developed. We observe the extensive presence of DCS-compatible ranges in real classifiers and more can be retrieved through our partitioning scheme. Second we divide the ranges in a classifier in terms of a hybrid utilization of various schemes. Technology advancement provides the necessary support for an open and flexible logical TCAM block division in order to avoid expensive hardware modifications and allow the use of DCS directly upon TCAM blocks.

Our scheme allows fast preprocessing, constant time searching, and dynamic incremental update. Theoretical analysis and simulation using synthetic classifiers show a substantial storage improvement using our scheme.

I. Introduction

The ever-increasing complexity of network applications and the tendency for low-cost equipment place additional pressure for high-performance packet classification which plays a significant role in network devices such as routers [1], [2]. Packet classification processes packets according to a collection of rules or classifier. Each rule individually determines an action, accept or discard for example, to apply to each packet that matches a rule [3]. The 5-tuple header fields (source/destination IP addresses, source/destination port values and protocol type) are the most prevalent matching criteria. A packet matches a rule only when all fields match. Ranges are often present in source and destination port fields to represent consecutive values such as $[1024, 65535]$ (both inclusive) or "greater than 1023". A port value matches a range if it falls within the range.

Ternary Content-Addressable Memory (TCAM) is increasingly used as the de facto industrial device for performing high-speed packet classification [4], [5]. A TCAM is an associative memory which can be viewed as an array of fixed-width entries. Each entry consists of ternary digits, each of which assumes the values 0, 1, or * ("don't care" or wildcard) [3], and stores data patterns in the form of (value, bit mask) pairs [6]. Given a packet, its searching key is produced using all 5-tuple header fields and compared simultaneously against all TCAM entries. Thus it takes $O(1)$ time to find the decision for any given packet [7], [8]. While TCAM-based approaches are fast and deterministic, they suffer from several disadvantages such as: inefficient representation of ranges, hardware cost, power consumption, and area constraints.

Prefix and exact values can be directly represented in TCAM entries while ranges cannot. Conventionally, the Prefix Expansion scheme, or PE [9], converts a range into a set of prefixes with each stored in a separate TCAM entry. The range expansion problem occurs when a rule with multiple range fields causes a multiplicative expansion in TCAM entries. For example $[1, 65534]$ stands for the worst-case range expansion scenario as represented by 30 TCAM entries using PE. If both port fields in a rule are $[1, 65534]$, it takes as many as 900 (30×30) TCAM entries to express the single rule! In general, a W-bit range may take up to $2(W-1)$ prefixes to represent [10]. Statistical analysis of real classifiers reveals that TCAM storage efficiency is greatly degraded due to a significant number of rules with port ranges and that number is steadily increasing [3], [4], [10]. Moreover TCAM memory inefficiency makes TCAM power consumption, area constraints, and hardware cost an even more serious concerns. For instance TCAM power consumption is mostly due to range expansion [6].

To address the problem our paramount goal is to improve TCAM storage efficiency by reducing the number entries required during range expansion. Although it is not feasible to optimally represent any range using one TCAM entry, some schemes [3], [4], [11], [12] substantially reduce the number of entries through smart encoding techniques. Besides storage improvement these schemes avoid hardware modification, which greatly impedes practical large-scale deployment and implementation, and hence allow fast preprocessing and incremental update.

We claim that our proposal, Divide and Conquer Scheme (DCS), performs better than previous range expansion schemes using encoding techniques due to the use of the divide and conquer methodology. We accomplish optimal representation for a particular class of ranges with each represented by exactly one TCAM entry. We define such a range $[2^p, 2^q - 1]$

with two conditions, $0 \leq p, q \leq W$ and $p + 1 < q$, as a DCS range. The other ranges are non-DCS ranges. DCS ranges such as $[1024, 65535]$ are widely present in real classifiers. Moreover a DCS range may be identified out of segments, or sub-ranges, partitioned from a non-DCS range. On the other hand a complementary scheme using encoding technique can easily work with ours to tackle non-DCS ranges. Owing to the hybrid utilization our scheme can achieve better range expansion than any single scheme using encoding technique as DCS ranges are always optimally represented.

Hybrid utilization are proposed by authors of [3], [4]. However no scheme using encoding technique achieves optimal expansion for an extensive class of ranges as we do for DCS ranges. In addition they do not justify the necessary conditions to enable hybrid utilization. Moreover practically implementing and deploying any range expansion scheme demands good performance in more essential metrics such as fast preprocessing, dynamic incremental update, and scalability to classifier size and IPv6.

As the first feasible solution, our DCS application architecture justifies such conditions. The core of the architecture is a grid of TCAM blocks. The number of blocks is determined by the maximum range dimension in a classifier. For example there are 4 blocks corresponding to 2 ranges in port fields of an IPv4 rule. Each block is associated with a single scheme or two hybrid schemes. If a rule contains any range in port fields, the range is identified by the DCS range partition strategy. DCS ranges are encoded using the DCS range encoding algorithm while a complementary scheme is responsible for encoding non-DCS ranges. Searching can be conducted in parallel with keys generated by scheme(s) applied in each block.

Our architecture is easy to implement and deploy as the three following conditions are met.

- Low-cost encoding resource. Extra unused bits in TCAM entries are available for range encoding purpose. Current TCAMs can support up to 133 million searches per second for 144-bit wide keys [13]. 104 bits are used to store an IPv4 rule and usually 4 more bits are reserved. As a consequence 36 additional unused bits are available and 32 bits are sufficient to encode two ranges.
- Hardware support. TCAM vendors have been providing mechanisms to address and activate smaller portions of the TCAMs, called blocks [14]. Our scheme requires logic division of TCAM entries into a set of blocks. TCAM chips can be flexibly divided to blocks in order to support the application of hybrid schemes. Meanwhile computation overhead caused by DCS key encoding is trivial owing to simple bit manipulation. Consequently our scheme can be easily implemented and deployed without hardware modification.
- Database independence. Our scheme does not depend on heuristic features in packet classifiers.

As a consequence the open and flexible architecture takes maximum advantage of each scheme for more efficient TCAM storage with trivial computation overhead and hardware cost. Thus fast preprocessing, dynamic incremental update, and good scalability are allowed. Theoretical analysis and simulation results using synthetic classifiers show our architecture substantially improves TCAM storage efficiency.

The remainder of this paper is organized as follows. In section II we present related work. In section III the details of our scheme are introduced. In section IV theoretical analysis and simulation using synthetic classifiers are described. Finally in section V we conclude our work.

II. Related Work

There are two fundamental packet classification solutions: software-based and hardware-based, primarily TCAM-based. Software-based solutions unanimously exploit various statistic characteristics of real classifiers to improve speed and memory requirements, and hence are not considered general [10]. In contrast TCAM-based solutions offer $O(1)$ searching time. To address range expansion problem, previous work falls into two primary categories with difference of changing TCAM hardware circuits.

Authors of [6], [15] present ideas for the problem by modifying TCAM circuits. For instance, van Lunteren et al. [15] proposed adding comparators at each entry to better accommodate range matching in classifiers. Ori Rottenstreich et al. [6] proposed a modified TCAM architecture that can use additional logic to significantly reduce the rule expansions, both in the worst case and when using real classifiers. Since TCAMs are complex devices and architectural changes that modify TCAMs involve millions of dollars of investment and more than two years of development time [4], these approaches are not easily to deploy in practice.

Approaches in [3], [4], [7], [10]–[13], [16], [17] do not change TCAM hardware. Our scheme follows this track. Authors of [4], [11] utilize unused bits in TCAM entries for range encoding. Authors of [3], [4] propose their own primary approaches with hybrid utilization of other complementary schemes. Authors of [10], [13], [16] instead intend to remove rule redundancy by means of a semantically equivalent set of entries in TCAM device for a packet classifier. However the range expansion problem still exists even though rule redundancy is tackled. Meanwhile the methodology makes preprocessing and update complicated and time-consuming. The approach in [17] deals with a portion of ranges, specifically disjoint ranges, and thus lacks database independence. Similarly authors of [12] only optimize 89% of all non-trivial rules in a real classifier.

Thus far no single approach performs optimally for any range type in every performance metric due to necessary tradeoffs. Nevertheless, these solutions are more easily adopted by networking device vendors because of no need for hardware modification. Our scheme deals with all kinds of ranges using hybrid schemes under the open and flexible DCS application architecture with database independence to classifiers.

III. DCS Design

Our scheme needs to preprocess a given classifier to store each rule into TCAM entries. The grid of blocks are established during the process. The DCS range partition strategy

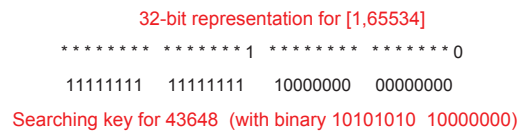

Fig. 1. Worst-case Range Expansion Scenario

reflects the divide-and-conquer methodology and differentiates DCS and non-DCS ranges. Our encoding algorithm expands DCS ranges and our key encoding algorithm generates search keys. These techniques will be introduced in a step-by-step manner in order to highlight underlying design choices. Finally we propose our DCS application architecture to support hybrid utilization of other schemes with ours. Suppose $[l, h]$ represent a range with low bound l and high bound h ($0 \leq l < h \leq 2^W - 1$). W denotes port width, e.g. 16 bits for an IPv4 rule. $[0, 2^W - 1]$ stands for the entire range spectrum.

3.1 Start Point: Worst-Case Scenario

Our first aggressive attempt is to optimally express $[1, 65534]$ using 32 bits.

Fig. 1 illustrates upper range representation for $[1, 65534]$ and lower searching key for a port value 43648. Every 8 bits are segmented by a space character for clear expression. In the range representation the first 16 bits represent the low bound generated by setting the least significant bit (rightmost) and putting others with wildcards. The high bound in the second 16 bits is constructed similarly with the only difference of clearing the least significant bit. The low and high bounds exclude 0 and 65535, respectively. The searching key for 43648 is created by concatenating its 16-bit OR bitmap and 16-bit AND bitmap. A W-bit OR bitmap for a given value is constructed by finding the leftmost 1 in the value and setting the remaining bits on its right. Likewise a W-bit AND bitmap is produced by finding the leftmost 0 and clearing the remaining bits on its right. For example, in Fig. 1, the leftmost 1 of 43648 is the most significant bit set and thus each bit is set in the OR bitmap. The leftmost 0 is the second significant bit and all the other bits on its right are cleared for the AND bitmap. The key matches as each bit in the key matches the corresponding bit in the range representation.

Compared to 30 entries using PE, the optimal representation saves 96.67% of the entries required to represent the range and will tremendously improve TCAM memory efficiency if the range $[1, 65534]$ is widely present in classifiers. Encouraged by the optimal representation, we desire to generalize it for other ranges.

3.2 Divide and Conquer

Since the low bound representation of $[1, 65534]$ excludes 0 by setting the least significant bit, we generalize the low bound representation of 2^k by setting the bit in the k^{th} least significant position (count from 0). Meanwhile we borrow the representation of $2^k - 1$ in the PE scheme [9] for high bound representation, e.g. 1023 expressed by setting consecutive bits from 0 to 9 when k is 10. Consequently optimal representation is obtained for DCS ranges. Algorithm 1 formulates the DCS range encoding algorithm to construct a $2W$-bit representation for a given DCS range by concatenating W-bit low bound and high bound.

Algorithm 1 DCS Range Encoding Algorithm

```
FuncEncode(p,q,W) {p and q satisfies p + 1 < q for DCS
range [2^p, 2^q − 1]; W: port width, 16 bits for IPv4.}
Initialize an empty string s; {generate W-bit low bound
using p}
for all i from W to 0 do
  if i = p then
    s.append('1');
  else
    s.append('*'); {'*': wildcard}
  i − −;
{generate W-bit high bound using q}
for all i from W to 0 do
  if i ≥ q then
    s.append('0');
  else
    s.append('*');
  i − −;
return s; {2W-bit result}
```

Given a port value, its search key is also established using $2W$ bits to match a DCS range. The first W bits of the key are constructed by finding the leftmost 1 in the value and setting the remaining bits on its right. The second W bits are a binary form of the value. Algorithm 2 formulates the algorithm.

Algorithm 2 DCS key encoding Algorithm

```
FuncKeyGen(v,W) {v: port value in binary; v_i: bit value (0
or 1) in position i of v; W: port width.}
Initialize an empty bit string k; {generate the first W bits}
for all i from W to 0 do
  if v_i = 1 then
    for all j from i to 0 do
      k.append('1');
      j − −;
    break;
  else
    k.append('0');
  i − −;
k.appendAll(v);
return k; {2W-bit searching key}
```

The optimal expansion remarkably enhances TCAM storage efficiency if DCS ranges are widely present in classifiers. The simplicity of bit manipulation makes key encoding for DCS ranges trivial computation overhead compared to most of other schemes using encoding techniques.

There are two categories of DCS ranges. Those naturally present in classifiers can be easily discovered by our range

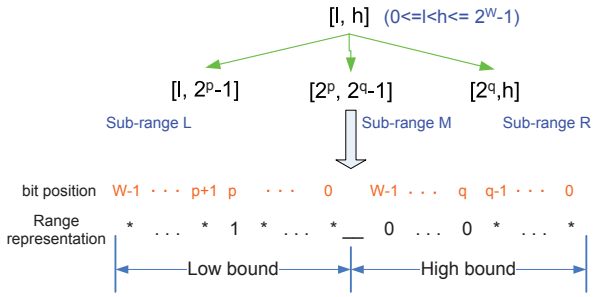

Fig. 2. DCS Range Partition and Encoding

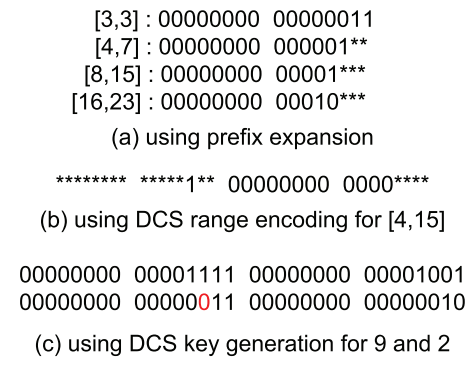

Fig. 3. DCS Range Encoding and Key Encoding Example

partition strategy without partitioning. To find more we try a partition for any non-DCS range and really perform the partition if a DCS range is found out of sub-ranges. According to the strategy a range $[l, h]$ ($0 \leq l < h \leq 2^W - 1$]) must fall into one of the following three cases.

1 No need to partition.
  - 1.1 A non-DCS range ($2^p < l < h \leq 2^{p+1} - 1$ and $0 \leq p < W$).
  - 1.2 Or a DCS range ($l = 2^p$, $h = 2^q - 1$ and $p+1 < q$).

2 Partition into two sub-ranges.
  - 2.1 $[l, 2^p - 1]$ and $[2^p, h]$ ($2^{p-1} \leq l < h < 2^{p+1} - 1$). Both non-DCS ranges.
  - 2.2 $[l, 2^q - 1]$ and $[2^q, h]$ ($l = 2^p$, $h < 2^{q+1} - 1$, $p+1 < q$); or $[l, 2^p - 1]$ and $[2^p, h]$ ($h = 2^q - 1$, $2^{p-1} < l$, $p+1 < q$). One DCS range and the other non-DCS range.

3 Partition into three sub-ranges $[l, 2^p - 1]$, $[2^p, 2^q - 1]$ and $[2^q, h]$ ($2^{p-1} < l \leq 2^p - 1$, $2^q \leq h < 2^{q+1} - 1$).
  - 3.1 All non-DCS ranges in case of $q = p + 1$.
  - 3.2 Or a DCS range $[2^p, 2^q - 1]$ among sub-ranges.

Note that in case 2.2 or 3.2 a non-DCS range is indeed partitioned. Otherwise we do not perform partitioning and only identify DCS ranges in case 1.2 or non-DCS ranges in other cases. Fig. 2 shows case 3.2 of the range partitioning. $[l, h]$ is partitioned into three sub-ranges in which sub-range M is a DCS range with W-bit low and high bound representations. In the low bound representation p bit is set and all other bits are wildcards. In the high bound representation total q bits (from 0 to $q - 1$) are wildcards and others are clear.

We summarize our design using an example in Fig. 3 to display how range partitioning, range encoding, and key encoding are performed upon $[3, 23]$. In (a) the range is expanded into 4 TCAM entries using PE. In (b) among three partitioned sub-ranges $[4, 15]$, a DCS range, is encoded into the 32-bit range representation. The 3^{rd} least significant bit is set as the low bound is 2^2. Likewise the least significant bits from 0 to 3 are set as the high bound is $2^4 - 1$. The other two sub-ranges are not shown. In (c) the key of 9 matches the DCS range representation of $[4, 15]$ while the key of 2 does not since its 3^{rd} bit (marked in red) is not set.

3.3 DCS Application Architecture

To support hybrid utilization of various range expansion schemes using encoding technique we propose an open and flexible DCS application architecture for better storage efficiency in TCAM devices. First we differentiate two classes of ranges when preprocessing a given classifier. Next a number of TCAM blocks are established according to range dimension in the classifier. Then a single scheme or hybrid schemes are applied to store rules with range representation in port fields into corresponding blocks. Eventually searching can be performed in parallel with trivial computational overhead. The purpose of the architecture aims to further improve TCAM memory efficiency and meet other essential measures.

The core of the architecture is a grid of 2^F number of blocks to accommodate a classifier with F ranges. For each range R_i ($1 \leq i \leq F$), $R_i{}^j$ ($1 \leq R_i{}^j \leq 3$) sub-ranges are engendered by applying DCS range partitioning. Eventually total $\prod_{i=1}^{F} R_i{}^j$ number of sub-ranges need to be distributed into the block grid. In an IPv4 classifier F is 2 regarding source and destination port fields and hence one complementary scheme is required for non-DCS ranges. Suppose "OP" represent the complementary scheme. The DCS block grid is divided into 4 blocks listed below. Any rule with at least one port range must fall in one of the blocks and we do not consider rules without range representation here.

1 b_{DCS} using DCS for both ports;
2 $b_{OP,DCS}$ using "OP" for source port and DCS for destination port;
3 $b_{DCS,OP}$ using DCS for source port and "OP" for destination port;
4 b_{OP} using "OP" for both ports.

Fig. 4 shows how a rule with source range $[126, 1025]$ and destination range $[59, 513]$ is processed in DCS block grid. In (a) $[126, 1025]$ is partitioned into three sub-ranges for two blocks, DCS range $[128, 1023]$ for b_{DCS} and the other two for b_{OP}. In (b) $[59, 513]$ is partitioned into three sub-ranges similarly. In (c) crossproducting these two sets of sub-ranges causes the distribution of the rule's 9 copies into 4 blocks. For example, $b_{OP,DCS}$ contains two copies with $[126, 127]$

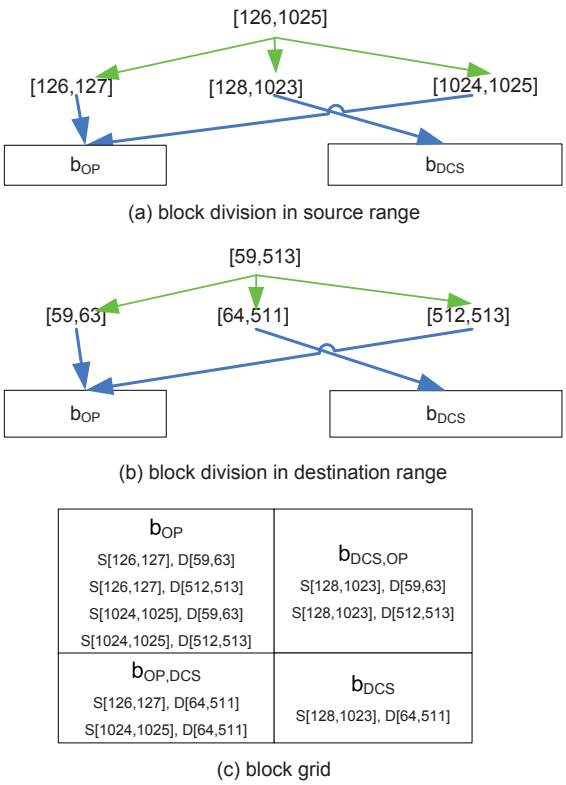

Fig. 4. DCS Block Division on Two Ranges

and $[1024, 1025]$ both encoded by "OP" in source port and $[64, 511]$ by DCS in destination port. If we adopt PE alone, the rule is expanded to 5 TCAM entries for each range and overall 25 entries are needed. Using hybrid schemes of DCS and PE the rule is expanded to 3 TCAM entries for each range and eventually only 9 entries are adequate to express the rule with 64% storage improvement in contrast to PE alone.

Given a port value, searching the matched range is performed in parallel upon all blocks. In each block, a specific search key is established using proper key encoding scheme(s). For instance on $b_{OP,DCS}$ the key is encoded using "OP" in source range and DCS in destination range.

IV. Evaluation

4.1 Theoretical Analysis

TABLE I refers to Table 4 in [4] and compares a variety of range expansion schemes in terms of four measures. Given a packet classifier with N rules and each rule has F range fields that are W-bit wide. These measures are (1) extra bits used; (2) capacity, the number of rules supported in the search subsystem; (3) update speed with respect to increasing demand for incremental update in high speed; (4) overhead on packet processor which highlights the cost of the extra logic needed. Those range expansion schemes besides DCS are Prefix Expansion (PE) [9], regional-based scheme and DIRPE [4]. Because of block division under DCS application architecture we analyze our scheme upon DCS block (with only DCS ranges) and discuss performance benefit from hybrid utilization.

First we examine extra bits used for encoding purposes. Our scheme requires W additional bits in TCAM entries for each DCS range. Consequently it takes W^F bits to encode F DCS ranges. As we discussed in section I, it is sufficient for DCS range encoding to exploit the unused 36 bits in most popular 144-bit TCAM entries for IPv4 classifiers. However, to represent an IPv6 rule at least 296 bits are demanded and the majority of bits are used for two 128-bit IPv6 addresses. Apparently the number of extra bits needed for 16-bit port representation is not a bottleneck. Although most popular TCAM entry widths, 144-bit and 288-bit, are not sufficient any more, 512-bit or customized TCAM width can be easily configured to obtain more unused bits for DCS range encoding.

Regardless of port dimension in b_{DCS} only one TCAM entry is enough to express the Cartesian product of all DCS ranges. Thus the worst-case capacity in b_{DCS} is 1. For other blocks with hybrid schemes or only complementary scheme, the worst-case capacity is entirely determined by the complementary scheme responsible for non-DCS ranges.

Dynamic incremental update is highly preferred by TCAM-based classification approaches and update efficiency fundamentally relies on the number of TCAM entries involved in an update operation. Due to the range expansion problem a rule may be stored in multiple TCAM entries and hence incremental update becomes more complex. Under DCS application architecture updating a range may involve at most three partitioned sub-ranges distributed in various blocks. Therefore $O(3^F)$ is the bound to update F ranges. For all other non-DCS ranges encoded by the complementary scheme, update efficiency is totally dependent on that scheme. Obviously both worst-case capacity and update efficiency for non-DCS ranges using the complementary scheme are the real bottleneck in overall performance of DCS application architecture.

The overhead on the packet processor is the actual searching complexity in generating keys. According to Algorithm 2 generating searching keys for F DCS ranges incurs only $O(2WF)$ bit operations, $2W$ for each range. Therefore it is not costly to have a limited number of additional gates with fairly simple logic needed for key encoding.

From what we have discussed so far it is fairly evident that our scheme allows simple and fast preprocessing, dynamic incremental update and scalability to classifier size and IPv6. As well it is easy to implement and deploy our scheme in real TCAM devices with such open and flexible application architecture.

4.2 Evaluation using Synthetic Packet Classifiers

Ten 300k-scale synthetic classifiers cover five firewalls (FW1 to FW5), four Access Control Lists (ACL1 to ACL4) and one IP Chain (IPC1) file from Classbench [18] seeds for evaluation. Although some of these classifiers have less than 300k rules due to its inherent parameter configuration, the scale and diversity of synthetic classifiers are adequately eligible for evaluation. Prefix Expansion (PE) is chosen as

TABLE I
COMPARISON OF KEY METRICS FOR RANGE ENCODING SCHEMES.

Metric	PE	Regional-based (r regions)	DIRPE (k-bit chunks)	DCS
Extra bits	0	$F(\log_2 r + (2n-1)/r)$	$F(W(2^k-1)/k - W)$	W^F
Worst-case capacity degradation	$2W - 2^F$	$2\log_2 r^F$	$2W/k - 1^F$	1
Cost of an incremental update	$O(W^F)$	$O(N)$	$O(W/k^F)$	$O(3^F)$
Overhead on the packet processor	None	Pre-computed table of size: $O(nF)$ comparators of width W bits	$O(W2^k/k)$ extra logic gates	bit manipulation of $O(2WF)$

the complementary scheme. Even if PE is highly inefficient to encode non-DCS ranges, we will demonstrate how storage performance is greatly promoted by means of our scheme. We evaluate and compare hybrid utilization, or "DCS&PE", with single scheme, or "Pure PE" according to Expansion Ratio (ERatio). ERatio is defined as the number of TCAM entries needed to represent a given set of ranges over the size of the range set.

TABLE II
RANGE STATISTICS IN PACKET CLASSIFIERS

Classifier	Src-Only	Dst-Only	Both	Total	Non-Range
FW1	5292	3890	17051	26233	195486
FW2	52661	0	0	52661	234028
FW3	3323	1746	13179	18248	189846
FW4	114255	22800	22491	159546	87659
FW5	3831	1921	9176	14928	187570
ACL1	0	13734	0	13734	171342
ACL2	0	73876	0	73876	224131
ACL3	0	65274	0	65274	224507
ACL4	0	27678	0	27678	272322
IPC1	7142	27533	0	34675	262828

TABLE II enumerates the range distribution in these ten classifiers. Each row presents the range distribution for a specific classifier. Each value in "Src-Only" indicates the amount of rules with only one range in source port for a specific classifier. Likewise "Dst-Only" for the amount of rules with only one range in destination port. Each value in "Both" indicates the amount of rules with ranges in both ports for a classifier and thus does not overlap with values in "Src-Only" and "Dst-Only". Each value in "Total" is the sum of values in "Src-Only", "Dst-Only" and "Both" for the same classifier. Each value in "Non-Range" is the amount of rules without any range representation. So the overall number of rules in a classifier is the sum of values in "Total" and "Non-Range". For example, FW1 classifier has total 221719 rules, the sum of 195486 rules in "Non-Range" and 26233 rules in "Total". Among the 26233 rules, there are 5292 rules with single source range, 3890 rules with single destination range and 17051 rules with double port ranges. In some classifiers, there is no range in one port such as values in "Src-Only" of ACL1 or "Dst-Only" of FW2. Consequently comparisons on single range do not involve all the classifiers.

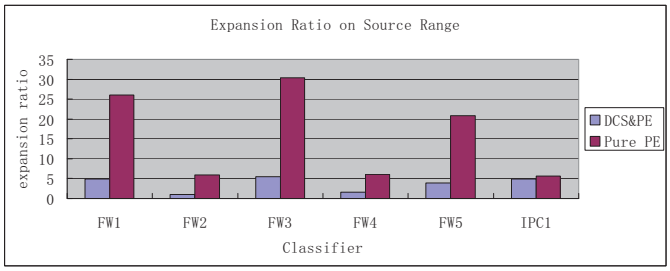

Fig. 5. ERatio Comparison on Only Source Range

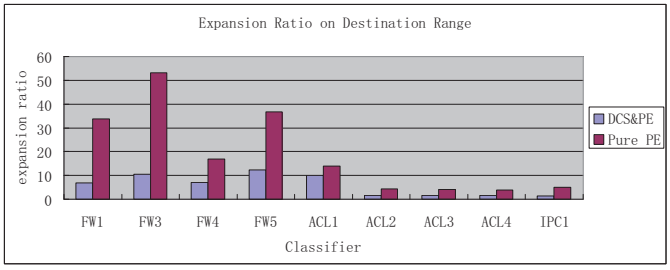

Fig. 6. ERatio Comparison on Only Destination Range

4.2.1 Comparison on Rules with Single Range: Fig. 5 demonstrates ERatios of "DCS&PE" and "Pure PE" for rules ("Src-Only" of TABLE II) with the only range in source port. For example in FW1 classifier 5292 rules with source-only range need 26308 and 137948 TCAM entries using "DCS&PE" and "Pure PE", respectively. Their ERatios are 4.97 and 26.07. We skip concrete entry numbers, e.g. 26308 and 137948 entries for FW1, and only compare ERatios due to paper space limit.

We notice three interesting points: (1) except in IPC1 ERatios of "DCS&PE" for all other classifiers are approximately 20% of corresponding ERatios of "Pure PE"; (2) in FW2 and FW4 the ERatios are nearly 1 because the majority of source-only ranges are $[1024, 65535]$, a natural DCS range; (3) The ERatios in IPC1 are quite close because most of its source-only ranges are non-DCS ranges, mainly small ranges such as $[22, 23]$. We thus observe a straightforward correlation between DCS ranges and ERatio of "DCS&PE" from these points: more DCS ranges in a classifier, both naturally present in classifiers and partitioned from non-DCS ranges, lead to smaller ERatio. It is not surprising at all since the discovery is consistent with optimal representation for DCS ranges.

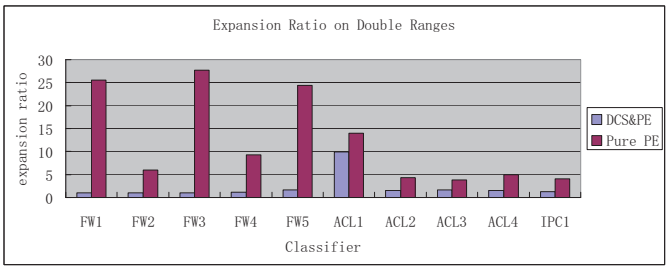

Fig. 7. ERatio Comparison on Either Range

Fig. 6 shows ERatios of "DCS&PE" and "Pure PE" for rules ("Dst-Ony" in TABLE II) with only range in destination port. For example in FW4 classifier 22800 rules with destination-only range need 163509 and 383684 TCAM entries using "DCS&PE" and "Pure PE" respectively. Their ERatios are 7.17 and 16.83.

For destination-only ranges the correlation between DCS ranges and ERatio of "DCS&PE" is applicable as well. Further observation upon range distribution in classifiers reveals another contributing factor for small ERatios: wide presence of "partial" DCS ranges satisfying only high bound, such as $[80, 1023]$ or ≥ 80 ($[80, 65535]$). Most of values within a partial DCS range can be included into a partitioned DCS range such as $[128, 65535]$ for ≥ 80 and hence the remaining values are contained in a small interval, such as $[80, 127]$, which is even represented using very few TCAM entries using inefficient PE. For instance these partial DCS ranges frequently occur in source port of FW2 rules, destination port of FW3 rules or both ports of FW4 rules and hence contribute to small ERatios for these classifiers using "DCS&PE".

4.2.2 Comparison on Rules with Either Range: Fig. 7 illustrates ERatios of "DCS&PE" and "Pure PE" for rules ("Total" of TABLE II) with at least one range. Note that if a rule has ranges in both ports, the number of TCAM entries needed to represent the rule is the Cartesian product of the numbers of entries needed for each range. For those rules with only one port range, the number of TCAM entries needed is precisely the number of entries for the single range. For example in FW3 classifier total 18248 rules with at least one port range need 18378 and 504858 TCAM entries using "DCS&PE" and "Pure PE". Their ERatios are 1.01 and 27.67 respectively.

We observe that in all the firewall classifiers except FW5 ERatios of "DCS&PE" are nearly 1 because of two reasons: (1) rules with double ranges are the majority; (2) $[1024, 65535]$ is the dominating range in these double-range rules ("Both" of TABLE II). In FW5 besides $[1024, 65535]$ there are a great number of partial DCS ranges. From the theoretical analysis we know a rule with DCS ranges in both ports only consumes one TCAM entry. Therefore the majority of rules with both DCS ranges definitely lead to extremely small ERatio as we see in all firewall classifiers. The rules in all the ACL classifiers do not contain any destination range and thus no crossproducting is needed. For IPC1 classifier there are tremendously few rules with DCS ranges in both ports. Nevertheless its ERatio of "DCS&PE" is better than that that of "Pure PE" since DCS ranges in single port still contribute to mild optimization.

The purpose to further improve TCAM storage efficiency necessitates introducing a complementary scheme for non-DCS ranges, primarily small ranges such as those in IPC1 classifier. Although an extensive collection of range expansion schemes using encoding technique are options on the table, a couple of schemes such as gray coding [3] or DIRPE [4] are better choices due to their excellent capability to encode small ranges and database independence. We have justified in section I that hybrid utilization is fully supported by meeting conditions such as hardware support. Our proposed open and flexible architecture allows the use of any complementary scheme using encoding technique for further storage efficiency, combined with optimal representation for DCS ranges.

4.3 Recursive DCS

Our scheme can be improved in a recursive manner. We examine an example $[1027, 1047]$ and discover its similarity to $[3, 23]$ if subtracting 1024 (2^{10}) in both bounds. What if we transform $[1027, 1047]$ to $[3, 23]$ under 2^{10}-bit mask and treat it as we partition and encode $[3, 23]$? We partition $[1027, 1047]$ into $[1027, 1027]$, $[1040, 1047]$ and $[1028, 1039]$ which is regarded as $[4, 15]$, a DCS range, and thus encoded using exactly one TCAM entry with 2^{10}-bit mask. We call the enhanced scheme Recursive DCS or RDCS. Apparently we need to extend the definition of DCS ranges and upgrade DCS application architecture with more blocks.

In general a non-DCS range $[l, h]$ is defined as m-bit DCS range if $[l - 2^m, h - 2^m]$ is a DCS range. In above example $[1028, 1039]$ is a 10-bit DCS range. When performing range and key encoding upon a m-bit DCS range, its $(W - m)$ (W is port width) most significant bits are masked. RDCS requires $O(W^2)$ blocks with each associated to a specific bit mask. Due to paper space limit we only illustrate RDCS using an example.

Fig. 8 shows an example of applying RDCS on a rule with two ranges. In (a) source range $[126, 1025]$ is partitioned into three sub-ranges for two blocks, e.g. $[128, 1023]$ for b_{16} (b_{DCS}) and the other two for b_{OP} which is the block using only complementary scheme. In (b) destination range $[32798, 33281]$ is partitioned into three sub-ranges for two blocks. $[32800, 33279]$ is a 15-bit DCS range for block b_{15}. In (c) four blocks are needed to contain all the copies of the rule. For example, $b_{16,15}$ is a block using DCS on source range and RDCS on destination range with 15-bit mask and accordingly contains one rule copy with a DCS range $[128, 1023]$ in source and a 15-bit DCS range $[32800, 33279]$ in destination. If we use "Pure PE" and "DCS&PE", the rule is represented using 30 entries (5 and 6 for source and destination) and 18 entries (6 and 3 for source and destination) respectively. Using "RDCS&PE" the number is only 9 entries (3 for both ranges).

Although intuitively RDCS has better storage efficiency than DCS, it has several drawbacks such as complicated

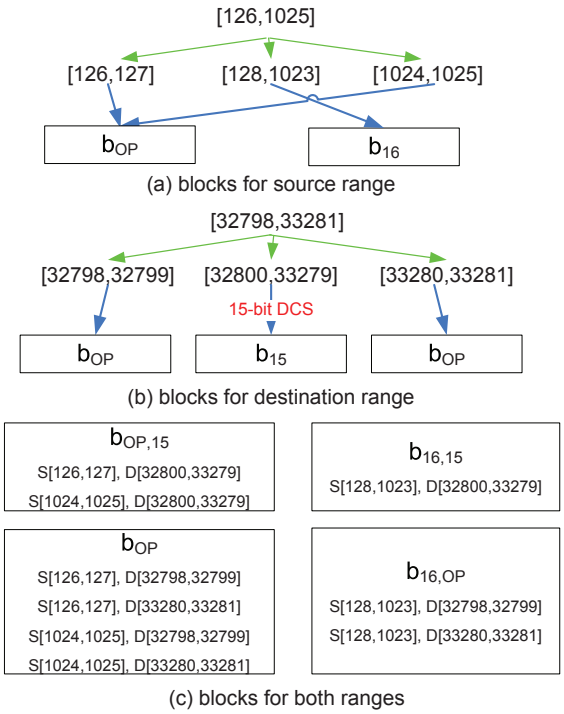

Fig. 8. Recursive DCS Example

preprocessing and update, non-trivial searching overhead due to key encoding, and nondeterminism in block division. In experiments using synthetic classifiers the overall encoding performance of RDCS is only slightly better than DCS due to scarcity of non-DCS ranges suitable for range partitioning using RDCS. After all besides storage efficiency we need to consider ease of implementation and deployment facilitated by other metrics such as fast proprocessing and simple update. Thus we need more real classifiers to evaluate RDCS and leave RDCS as our future work.

V. Conclusion

As the first scheme using divide-and-conquer to address range expansion problem which sharply degrades storage performance for TCAM-based packet classification approaches, our proposal presents the first practical application architecture for hybrid utilization of various schemes for further TCAM storage improvement. According to our scheme any range is either optimally represented by exactly one TCAM entry, or expressed using another best suitable scheme. Therefore TCAM storage performance is further improved. Under DCS application architecture, TCAM entries can be flexibly divided into blocks upon which searching in constant time can be performed in parallel. Simplicity of block division and encoding methods using available unused bits in TCAM entries as well as trivial computation overhead in searching lead to ease of practical implementation and deployment of our scheme without hardware modification. Theoretical analysis and simulation using synthetic packet classifiers justify our architecture's excellent performance in fundamental metrics such as storage efficiency, preprocessing, update, and scalability.

References

[1] P. Gupta and N. McKeown, "Classifying packets with hierarchical intelligent cuttings," *IEEE Micro*, vol. 20, no. 1, pp. 34–41, 2000.

[2] Y. Qi, L. Xu, B. Yang, Y. Xue, and J. Li, "Packet classification algorithms: From theory to practice," in *Proceedings of IEEE INFOCOM*, Apr. 2009, pp. 648–656.

[3] A. Bremlerr-Barr and D. Hendler, "Space-efficient TCAM-based classification using gray coding," *IEEE Transactions on Computers*, vol. 61, pp. 18–30, Jan. 2012.

[4] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 193–204, Oct. 2005.

[5] Y. Sun and M. S. Kim, "Tree-based minimization of TCAM entries for packet classification," in *Proceedings of the 7th IEEE Consumer Communications and Networking Conference*, Jan. 2010.

[6] O. Rottenstreich, R. Cohen, D. Raz, and I. Keslassy, "Exact worst case TCAM rule expansion," *IEEE Transactions on Computers*, vol. 62, pp. 1127–1140, Jun. 2013.

[7] Y. Sun and M. S. Kim, "Bidirectional range extension for TCAM-based packet classification," in *Proceedings of the 9th International IFIP TC 6 Networking Conference*, May 2010, pp. 351–361.

[8] ——, "A hybrid approach to CAM-based longest prefix matching for IP route lookup," in *Proceedings of IEEE GLOBECOM*, Dec. 2010.

[9] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 191–202, Oct. 1998.

[10] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, "Packet classifiers in ternary CAMs can be smaller," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 311–322, Jun. 2006.

[11] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES: Dynamic range encoding scheme for TCAM coprocessors," *IEEE Transactions on Computers*, vol. 57, pp. 902–915, Jul. 2008.

[12] O. Rottenstreich, I. Keslassy, A. Hassidim, H. Kaplan, and E. Porat, "On finding an optimal TCAM encoding scheme for packet classification," in *Proceedings of IEEE INFOCOM*, Apr. 2013.

[13] A. X. Liu, C. R. Meiners, and Y. Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *Proceedings of IEEE INFOCOM*, Apr. 2008.

[14] Y. Ma and S. Banerjee, "A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification," *ACM SIGCOMM Computer Communication Review*, vol. 42, pp. 335–346, Oct. 2012.

[15] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 560–571, May 2003.

[16] A. X. Liu, C. R. Meiners, and E. Torng, "TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 490–500, Apr. 2010.

[17] A. Bremler-Barr, D. Hay, and D. Hendler, "Layered interval codes for TCAM-based classification," in *Proceedings of IEEE INFOCOM*, Apr. 2009.

[18] D. E. Taylor and J. S. Turner, "ClassBench: A packet classification benchmark," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 499–511, Jun. 2007.