A Novel Rule Mapping on TCAM for Power Efficient Packet Classification

S. M. SRINIVASAVARMA VEGESNA, ASHOK CHAKRAVARTHY NARA, and NOOR MAHAMMAD SK, IIITDM Kancheepuram, India

Packet Classification is the enabling function performed in commodity switches for providing various services such as access control, intrusion detection, load balancing, and so on. Ternary Content Addressable Memories (TCAMs) are the de facto standard for performing packet classification at high speeds. However, TCAMs are highly costlier both in terms of cost and power consumption, forcing the switch vendors towards placing lots of effort for power management. Hence, power-efficient solutions for TCAM-based packet classification are highly relevant even today. In this article, we propose a novel rule placement algorithm based on the unique field values' presence within the rule databases. We evaluate the total search that is needed to be inspected with respect to the traditional placement approach and the proposed placement approach based on the information content within the fields. Simulation results showed an average reduction of 30.55% in the search space by the proposed placement approach, thereby resulting in an average reduction of 18.85% per search energy over TCAM. With typical TCAM clock speeds ranging between 200–400MHz, this reduction in the per-search energy maps to a huge reduction in the total energy consumed by the TCAM-based network switches. The proposed solution is plug-and-play type requiring only minimal pre-processing within the Network Processing Unit (NPU) of the switches and edge routers.

CCS Concepts: • **Networks** → **Packet classification**; *Routers*; • **Hardware** → *Networking hardware*;

Additional Key Words and Phrases: Packet classification, Ternary content addressable memories, power efficient switches and routers, rule database

ACM Reference format:
S. M. Srinivasavarma Vegesna, Ashok Chakravarthy Nara, and Noor Mahammad Sk. 2019. A Novel Rule Mapping on TCAM for Power Efficient Packet Classification. *ACM Trans. Des. Autom. Electron. Syst.* 24, 5, Article 48 (June 2019), 23 pages.
https://doi.org/10.1145/3328103

1 INTRODUCTION

Packet classification involves classifying the incoming packets by comparing the packet header contents against a set of pre-defined rules. Commodity switches and routers employ packet classification to provide various services such as access control, firewall screening, intrusion detection, differentiated services, traffic billing and shaping, and so on (Baboescu and Varghese 2001).

The work is supported by the Ministry of Electronics and Information Technology, Govt. of India, under Visvesvaraya Ph.D scheme.
Authors' addresses: S. M. S. Vegesna, A. C. Nara, and N. M. Sk, IIITDM Kancheepuram, Chennai, 600127, India; emails: {coe16d001, coe14b024, noor}@iiitdm.ac.in.

1084-4309/2019/06-ART48 $15.00
https://doi.org/10.1145/3328103

Table 1. Sample Rule Database

Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
192.151.11.17/32	15.0.120.4/32	0 : 65535	1221	TCP	Allow
192.151.11.17/32	15.0.120.4/32	0 : 65535	0 : 1599	UDP	Deny
0.0.0.0/0	204.152.188.19/32	750	0 : 65535	*	Deny
0.0.0.0/0	204.152.184.101/32	*	80	*	Alert

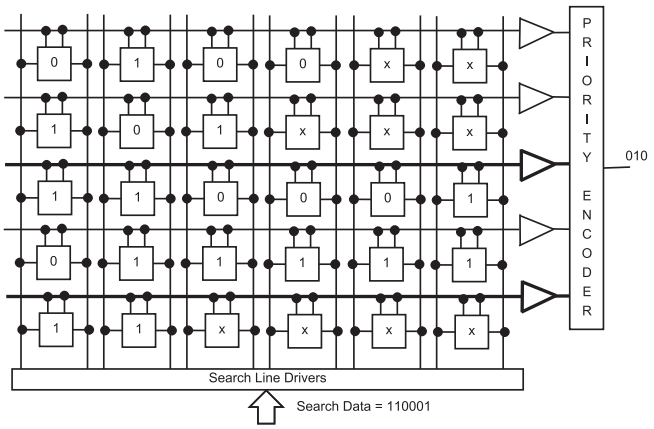


Fig. 1. TCAM structure for packet classification.

With ever-growing rule database sizes and enhanced optical backbone networks coming into the use, there is an increasing gap between the router's classification ability and the backbone speeds (Zheng et al. 2006). Hence, the study of efficient packet classification solutions is still relevant. An example of packet classification rule database is shown in Table 1. Typically packet classification requires inspection to be done on five fields: Source and Destination IP addresses, layer four ports, and protocol field. The longest prefix match is done for IPs, whereas ports require range match and exact match is done for the protocol fields. Traditionally, packet classification techniques can be classified into three broad categories based on the implementation platforms: (1) Software, (2) Hardware/ASIC, and (3) Reconfigurable/FPGA-based solutions (Taylor 2005). Software-based solutions offer better flexibility, but their performance cannot exceed beyond 10Gbps (OC-192) even after implementing them on the state-of-the-art platforms (Qi et al. 2009; Taylor 2005). Hence, hardware-based solutions using Ternary Content Addressable Memories (TCAMs), which is shown in Figure 1, have become the defacto solution for high speed packet classification in the core routers (Meiners et al. 2011b; Van Lunteren and Engbersen 2003; Zheng et al. 2006). But the TCAMs advantage comes with a high cost, low chip density, and higher power consumption (Agrawal and Sherwood 2008). Field Programmable Gate Arrays (FPGAs)-based solutions have gained importance recently, as they offer a compromise between flexibility and performance (Jiang and Prasanna 2009, 2012; Song and Lockwood 2005). But using NPUs for FPGA update should be able enough to generate the bit maps faster, which makes the update process more computationally intensive (Song and Lockwood 2005). With Graphic Processing Units (GPUs) gaining huge relevance in speed-critical applications, GP-GPU-based packet classification is gaining importance (Varvello et al. 2016). But the GP-GPUs are not stand-alone devices and involve higher cost. Hence, the GP-GPU-based packet classification is usually preferred only in extremely demanding situations.

1.1 Motivation

The power-consumption issue in TCAMs forces many vendors towards providing redundant power systems. Also, in any typical Network Processing Unit (NPUs), TCAMs stand as the largest power-consuming element. This is due to the TCAM circuit structure. A TCAM, shown in Figure 1, can accommodate 0, 1, and ∗ (don't care) directly, unlike traditional Static Random Access Memories (SRAMs) and Binary Content Addressable Memories (Bi-CAMs). A single cell of TCAM contains typically 12–16 transistors, whereas a single Bi-CAM cell is made up of 8 transistors and SRAMs contain only 6 transistors per cell (Agrawal and Sherwood 2008; Liu et al. 2016). The additional transistors in the TCAM were to accommodate don't care condition. Hence, TCAMs though promising in terms of performance, have to be effectively utilized for achieving better power and delay performance. In addition, the match-line unit in the TCAM is complex, involving matchline pre-charge (before the launch of every search operation), matching, and finally the Matchline Sensing Amplifier (MLSA) for driving the priority encoder. The launch of each search operation begins with pre-charging all matchlines to high. Then search operation will result in retaining the pre-charge value of those matchlines where every bit is a match against the input and remaining lines will discharge to ground. The MLSA will detect the matched and mismatched lines, thereby driving the encoder. This happens for all the matchlines in the TCAM for every input, and this unit stands as the significant contributor of the total TCAM power consumption. Optimizing the search process with respect to matchline circuitry can significantly contribute to the power reduction of the TCAM search.

1.2 Proposed Solution

In this work, we propose a novel placement algorithm for mapping the appropriate fields in the rule database on the appropriate TCAM segments. The algorithm takes into account the presence of unique field values in each field for finding out the placement order on TCAMs. In five-field packet classification, rather than classifying based on the entire string at a single stretch, the advantage of chunk-based matchline activation in the TCAM is utilized (Agrawal and Sherwood 2008; Meiners et al. 2011b; Pagiamtzis and Sheikholeslami 2004). The total string is separated based on the individual fields. Packet classification remains independent of the field that gets classified first. It means, using If structure, you can classify any field at the outer most loop and any field in the innermost loop and the classification remains unaffected by this. The algorithm takes the rule database as an input and gives out the fields in the sorted order of the number of distinct field values present in them. The NPU then maps these fields onto the TCAM by placing the field with the highest distinct field count at the left-most segment and the field with the lowest count in the right-most segment. Using these two (Field Segmented Matchline (ML) Activation and Proposed field order placement) guarantees the maximum purity in the first chunk of the TCAM and this happens recursively. The proposed solution is mapped with the state-of-the-art range-encoding solutions available in the literature. The performance of the proposed mapping is compared against the traditional mapping in terms of search space. The reduction in the search space is again mapped to the TCAM power using the TCAM Power and Delay model (Agrawal and Sherwood 2008), and the efficacy of the proposed solution in terms of per-search energy is thereby evaluated.

1.3 Paper Organization

In Section 2, we provide the literature review. TCAM-based packet classification issues and TCAM device analysis is also discussed in Section 2. The proposed placement algorithm is discussed along with an example in Section 3. Also in Section 3, the complexity analysis and incremental update solutions are discussed in detail. The impact of the proposed solution in terms of search space and

TCAM *energy per search* is tested against real-life databases and presented in Section 4. The work is summarized and concluded in Section 5.

2 BACKGROUND

Packet classification is a well-studied problem both algorithmically and architecturally. But no work has been reported in the literature in terms of rule field placement order on TCAMs or Bi-CAMs. In this section, we review various classification techniques in the literature and provide a brief overview on the software-based approaches. Later, we elaborate the TCAM-based classification in detail.

Though classification algorithms have been studied for a long time, applying them strategically for packet classification was proposed by Gupta and McKeown (2000), in which the structural properties of rule sets were studied by employing heuristics for achieving efficient memory-speed trade-off. Hi-Cuts (Gupta and McKeown 2000) employs an intelligent trie data structure for classification in which a localized decision at each node is taken. Each node in the trie is associated with the number of cuts that has to be made, the cut direction, and the number of rules that have to be associated with each leaf node. HyperCuts algorithm (Singh et al. 2003) and HyperSplit (Qi et al. 2009) can be seen as extensions to the Hi-Cuts algorithm with variations at the node attributes such as the binth size (number of nodes associated with the leaf node), the number of root nodes, and the localized information attributes utilized at each node. The above algorithms fall into the category of Decision Tree approaches. The major disadvantage is that their implementation has the least chance for parallelization and also does not suit hardware implementation. Recursive Flow Classification (RFC), proposed by Gupta and McKeown (1999), carries the classification at the individual field levels, thereby successfully merging the independent results in the later stages and arrives at the final result in the last stage. RFC suits properly for hardware-based implementation, whereas Hi-Cuts fits for software implementation. RFC, and its efficient extension Hierarchical Space Mapping (HSM) (Xu et al. 2005) falls into the decomposition-based algorithms. These algorithms can be well accelerated using SRAM memories and hence can deliver higher look-up rates. The primary disadvantage with both RFC and HSM is that their memory requirement is exponential and hence costlier. D. E. Taylor et al. proposed Distributed Cross-Producting of Field Labels (DCFL) (Taylor and Turner 2005), in which labels were associated with the unique values in each field. Meta-Labels were associated with the field cross-products and these meta-labels were searched using bloom filters at the respective stages, thereby producing the final classification result. Unlike RFC, DCFL yields better memory efficiency at the cost of reduced speed and increased complexity. Even though highly flexible in terms of rule database size and dimension, the algorithmic approaches cannot scale for speeds beyond 10*Gbps* even though they were implemented on the Multi-Core state-of-the-art platforms (Qi et al. 2009; Taylor 2005). Techniques employing hashing (Papaefstathiou and Papaefstathiou 2007; Xu et al. 2014) improve upon the above algorithmic approaches in terms of both memory and performance as well. However, with link speeds of up to 100*Gbps* currently under deployment, the practicality of software-based classification is highly limited and hardware-based classification employing TCAMs remains as an alternative for speed-critical applications.

2.1 TCAMs for Packet Classification

Ternary Content Addressable Memories (TCAMs) are the de facto industry standard for high-speed packet classification. TCAMs perform an exhaustive search against all the entries simultaneously and yield the final best match result with the help of a priority encoder. But TCAMs are extremely power-hungry and also costlier. When compared with the Static Random Access Memories (SRAMs), the scaling up of the clock frequency in TCAMs has been significantly lower

Table 2. State-of-the-Art Range Encoding Solutions

Dimension	Deterministic			Non-Deterministic	
	Binary	SRGE	Optimal	PPC (Van Lunteren and Engbersen 2003)	External (Rottenstreich et al. 2016)
1	$2w-2$	$2w-4$	w	1	–
2	$(2w-2)^2$	$(2w-4)^2$	$2w$	1	$4w-3$
d	$(2w-2)^d$	$(2w-4)^d$	dw	1	–

(Agrawal and Sherwood 2008). This can be attributed to their extremely high power consumption. With low switching clock speeds, though TCAMs perform the classification in $O(1)$ time, the net throughput is limited. The degradation of the TCAM performance can be directly related to the increase in the TCAM size or the number of TCAM bits taken to accommodate a classifier. Hence, the existing solutions in the literature either directly or indirectly aim to reduce the TCAM size required for a given classifier. The existing solutions for TCAM-based packet classification can be divided into three major categories: (1) Classifier Compression, (2) Range Encoding, and (3) Indexed TCAMs. Classifier compression is a popular way to optimize the TCAM usage. It is achieved either through redundancy removal (Liu et al. 2008) or using equivalent transformation techniques (Liu et al. 2010; Meiners et al. 2011a). In equivalent transformation, the given classifier is converted into a smaller classifier that is semantically equal. In contrast, Wei et al. (2016) use block permutations on the given classifiers to obtain the non-equivalent classifiers and achieve better compression than the above transformation approaches.

The major challenge that stands in the middle of using TCAMs for packet classification is Range to Ternary Conversion. Source port and Destination port fields in packet classification rule databases are specified in terms of ranges, like 0–30,000. TCAMs cannot accommodate ranges within them directly, and hence these ranges have to be converted to Ternary strings. Converting a range to a ternary value using binary encoding results in $(2w-2)^d$ ternary strings (Liu 2002), where d is the dimension and w is the number of bits used to represent a range. This phenomenon in packet classification is termed as *rule blow-out,* and this rule blow-out results in memory inefficiency for the TCAM-based packet classification. To address this, various encoding schemes were proposed and the state-of-the-art solutions along with their upper bounds are shown in Table 2. In Table 2, the solutions were categorized into deterministic and non-deterministic solutions. In the case of deterministic solutions, after the placement of encoded string on the TCAM, the packet header can be fed directly for classification. But in non-deterministic encoding, packet headers have to be pre-processed every time they are applied to TCAM for look-up. Non-deterministic encoding techniques outperform deterministic encoding in terms of reduction in the rule blow-out. But their inability to integrate with the existing hardware due to the requirement of header pre-processing and inability to support incremental updates limit their applicability. In Table 2, three deterministic solutions—Binary Encoding that is a naive prefix encoding, Short Range Grey Encoding (SRGE) (Bremler-Barr and Hendler 2012), and Optimal Encoding (Rottenstreich et al. 2013)—are shown with their complexities. Optimal Encoding entails a conversion complexity of $d \times w$, and it is the most efficient scheme in terms of conversion efficiency. Also, in Rottenstreich et al. (2013), a lower bound proof was given to prove that $d \times w$ is the lower bound that could be achieved in deterministic form. The proposed solution seamlessly integrates with any of the deterministic encoding solutions shown in Table 2. In this work, we map our solution to all three deterministic encoding schemes and perform the power-efficiency analysis on TCAM. Apart from

encoding schemes, other works in the literature provide architectural modifications to the existing TCAMs for supporting range matching directly without expansion (Murugesan and Sk 2017; Spitznagel et al. 2003). However, these solutions are mostly based on custom hardware and involve more development cost and time.

Indexed TCAMs divide the total classifier/rule database into multiple subsets and place each subset in an individual TCAM block. Rules placed in each subset are supplemented with an identifier, and the identifiers corresponding to various TCAM blocks are placed in a pre-classifier. PC-Trio (Banerjee et al. 2015), SmartPC (Ma and Banerjee 2012), and GreenTCAM (Li et al. 2016) fall into this category. The major objective of the indexed TCAMs is to reduce the power consumption by activating only a small set of TCAM entries and this activation is governed by the indexed TCAM entries. The size requirement for the total classifier almost remains the same and may even increase in some cases due to redundancies across various TCAM blocks. PC-Trio (Banerjee et al. 2015) uses three TCAMs out of which two are indexed and the other one is a general TCAM. Almost all the indexed TCAMs contain a general TCAM that is activated for all packet inputs. The delay improvement and power improvement in PC-Trio is claimed based on the fact that the two indexed TCAMs do not require any priority encoder. However, commodity TCAM chips come with priority encoders (Xu et al. 2014). PC-Trio integrates wide SRAMs for addressing range encoding and also for reducing the total number of classifiers stored in the TCAMs. A special logic circuitry is used to compare the inputs stored in the wide SRAM cells. PC-Trio, though, uses TCAMs and demands custom logic design at various levels in the architecture, including TCAMs without priority encoders. This affects the deploy ability of the PC-Trio in the existing TCAM-based classifier frameworks and incurs huge development cost and time. Also, the improved search latency achieved in PC-Trio by eliminating the priority encoder in indexed TCAMs will be affected by the general TCAM whose search result is also needed for making the final classification design. SmartPC (Ma and Banerjee 2012) constructs a two-dimensional pre-classifier, and the constructed pre-classifier is searched first before activating only a few specific blocks in the main classifier. The power savings in the SmartPC were obtained based on assuming 50 independent classifier blocks. Having such a huge number of TCAM blocks may not be a possibility in practice. The block sizes were estimated in SmartPC without considering any rule expansion for every rule that is not the case in packet classification. Green TCAM (Li et al. 2016) improves upon SmartPC by using two separate 1-Dimensional pre-classifiers as opposed to a single 2-Dimensional pre-classifier used by SmartPC. This approach will result in reduced general TCAM rules. The TCAM-based solutions Mn-Mx approach (Faezipour and Nourani 2009) and DPPC-RE (Zheng et al. 2006) also use indexing. But their architectures' primary motive was not the power savings. Mn-Mx uses indexing for yielding multi-match results from the TCAM. DDPC-RE uses indexing for achieving parallel packet classification to reach a throughput of 100*Gbps* and beyond.

This article proposes a novel rule field placement approach through which the search space on the TCAM can be recursively reduced, resulting in search power reduction. To the best of our knowledge, this is the first work that uses field placement for optimizing TCAM search power and hence does not fall into any of the three categories of TCAM-based packet classification that have been mentioned. The proposed approach fits with the existing TCAM classifier frameworks without any hardware overhaul requirement. Almost all the indexed TCAM solutions demand custom hardware modifications and additional logic circuitry. This can limit their adaptability within the existing switches/routers. The proposed placement approach only requires pre-processing modifications in the line card or the NPU to determine the field placement positions in the respective chunks of the TCAMs. *Also, the proposed field placement can be integrated with the existing indexed solutions seamlessly for achieving further power savings.* The computationally less intensive

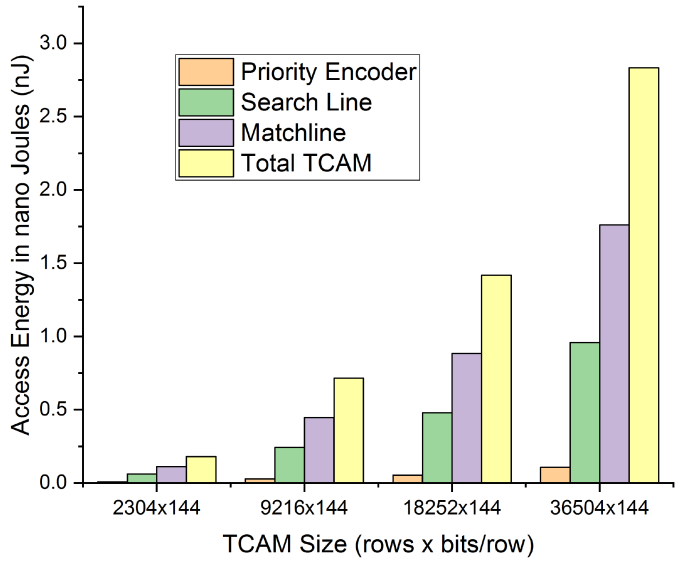


Fig. 2. TCAM energy per-search consumption for various sub-circuits.

pre-processing supports the easy integration of the proposed placement technique with other solutions for obtaining the further optimization in the TCAM search process.

2.2 TCAM Power Behavior

Though TCAMs provide $O(1)$ lookup time, the major disadvantage is their extremely high power consumption. Any approach that improves the TCAM look-up energy can significantly reduce the switch/router cost, as larger TCAMs require redundant power systems and cooling arrangements for subsiding the heat generated from heavy power consumption in the large-scale switches (Agrawal and Sherwood 2008). Also, TCAMs perform very poorly in terms of search delay and energy when compared with an equivalent SRAM. A single search on $1Mb$ TCAM takes $27.61ns$ per access and $0.715nJ$ per search, while a same size SRAM takes only $0.818ns$ per access and $0.092nJ$ per search as per the TCAM (Agrawal and Sherwood 2008) and SRAM (Muralimanohar et al. 2009) power and delay models. This means TCAM is almost 30 times and 7 times costlier than SRAM in terms of performance in time and energy, respectively. In a TCAM, the three primary energy-consuming and delay-causing elements are: (1) Matchline (ML), (2) Searchline (SL), and (3) Priority Encoder (PE) per (Agrawal and Sherwood 2008). Figure 2 shows the energy per-search consumption of various TCAM configurations with respect to the MatchLine, Searchline and Priority Encoder. The matchline contributes the highest amount of power that is, on average, equal to 62.35% of the total power. The searchline and priority encoder come second and third, respectively. In this work, *the reduction in the search space achieved through the modified rule field placement will map directly to the reduction in the total number of matchline segments being activated. This reduction in the matchline activation will reduce the TCAM energy per search significantly, as matchline contributes to the significant portion of total TCAM per search energy.*

3 PROPOSED APPROACH

TCAMs act as the co-processors within the router and assist the Network Processing Unit (NPU) in performing the High Speed Look-up, as shown in Figure 3. A TCAM look-up can happen either as a single string obtained by concatenating the individual fields together or at the field level segmented

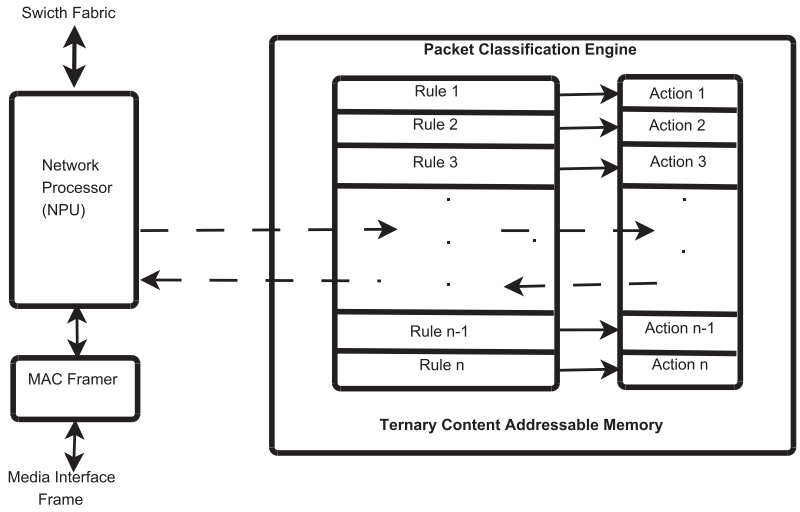


Fig. 3. Router/switch with TCAM as co-processor.

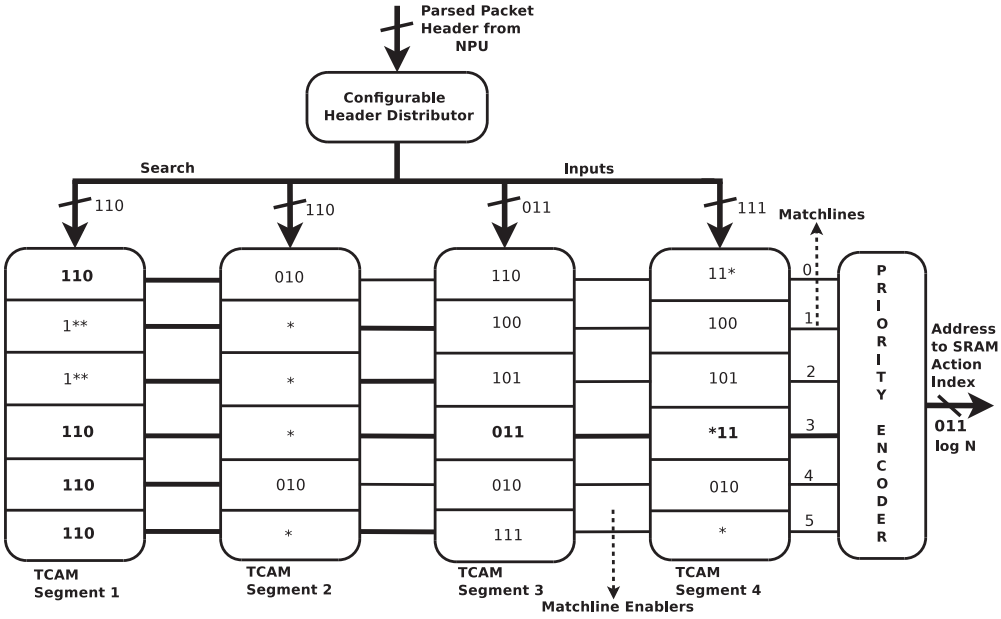


Fig. 4. Matchline segmented TCAM architecture with naive placement.

manner. The case of single-stretch look-up is shown in Figure 1. However, single-stretch look-up will require the entire matchline to be activated and thereby result in huge power consumption. The alternative for this is the segmented matchline look-up (Pagiamtzis and Sheikholeslami 2004), which can reduce the power consumption by a huge factor. This is done by pruning out the non-matched lines in the previous segment from pre-charging in the successive segments. An example of a matchline-based segmented TCAM classification engine for the database given in Table 3 is shown in Figure 4. As shown in Figure 4, a $segment_i$ will activate only those matchlines whose corresponding lines in the $segment_{i-1}$ is a match. The enable lines corresponding to the matched

Table 3. Rule Database

F1	F2	F3	F4	Action
110	010	110	11*	Accept
1**	*	100	100	Deny
1**	*	101	101	Log
110	*	011	*11	Log
110	010	010	010	Deny
110	*	111	*	Accept

Table 4. Input Traces for Classification

Flow	Header	Packets/Flow
H1	<111, 110, 101, 011>	10
H2	<111, 001, 010, 011>	15
H3	<110, 011, 110, 101>	5
H4	<011, 010, 001, 100>	10
H5	<100, 101, 110, 111>	7
H6	<010, 011, 111, 111>	13

locations for the inputs fed to each segment were highlighted in Figure 4. For every input, *the matchline units corresponding to all the entries (six in Figure 4) in the left-most/first filtering segment* will be activated and the pruning will happen only from the second segment onwards. As shown in Figure 4, for an input header tuple <110, 110, 011, 111>, the total number of matchline units activated in the various segments was 17. The break-up for 17 is as follows: All the matchline units corresponding to the 6 entries of the first segment are activated. This is followed by 6, 4, and 1 matchline units activation in the second, third, and fourth segments, respectively, as shown highlighted in Figure 4. Though this is better compared to the single-string-based approach, the pruning in the matchline units activation is not optimal. In this work, we propose an efficient heuristic-based approach based on the field placement on the TCAM segments to achieve efficient pruning of the total number of matchline units getting activated.

In Section 3.1, we formally define a few important definitions; detailed discussion of the proposed design is presented later. In Section 3.3, the analysis of the proposed design with respect to pre-processing, update, and the adaptability of the solution to the off-the-shelf TCAMs is presented.

3.1 Optimal Field Placement

A Rule database (S) is a collection of total number of rules (N) against which each incoming packet header has to be matched. For each incoming packet, once the header is extracted, only required fields have to be fed to the classifier and the best match rule will be the output delivered by the classifier. Then a necessary action as dictated by the best match rule will be taken by the switch.

Rule: A rule R is defined as a tuple containing D fields. An i^{th} component of a rule is termed as $R[i]$. An incoming packet P is said to have matched a rule R, if $\forall i$; $P[i]$ matches $R[i]$. The field i determines the kind of matching that has to be performed. In any packet classification problem, be it traditional 5 field or 12 field Openflow (McKeown et al. 2008), three different kinds of matching have to be done. They are: (1) Longest Prefix Match on IP Fields, (2) Range Matching on Port Fields, and (3) Exact Match on protocol, flag fields, and MAC address.

Range: W is the number of bits used to represent either the lower or higher value of a range. L gives the lower value of the range, and H gives the higher value of the range. Any $W - bit$ range S given as $S = [L : U]$ takes the range from $0 : 2^w - 1$ and the conditions $0 \leq L \leq 2^w - 1$ and $0 \leq H \leq 2^w - 1$ and $L \leq H$ are satisfied.

Problem Definition: Given a Rule Database S having N rules and F fields ($d = F$) and an accordingly segmented TCAM with F segments. Choosing an appropriate TCAM segment S_i for the field F_i to achieve least number of matchlines getting enabled in each segment of the TCAM is the problem to be addressed.

Table 5. Field Information Properties of ClassBench Rule Databases

Database (Rule Count)	# of Distinct Field Values				
	SA	DA	SP	DP	Pr
ACL 100 (98)	30	61	1	39	4
ACL1 (752)	95	202	1	140	4
ACL 1K (916)	103	297	1	99	4
ACL 5K (4,415)	805	640	1	108	4
ACL 10K (9,603)	4,569	726	1	108	4
FW 100 (92)	12	25	11	26	5
FW1 (269)	57	66	13	43	5
FW 1K (791)	124	175	13	42	5
FW 5K (4,653)	1746	2,714	13	43	5
FW 10K (9,311)	3,638	6,951	13	43	5
IPC 100 (99)	63	63	14	14	4
IPC1 (1,550)	152	128	34	54	7
IPC 1K (938)	336	436	28	44	6
IPC 5K (4,460)	585	1,251	34	53	7
IPC 10K (9,037)	1,515	2,726	34	54	7

Field Selection Method: Choosing a methodology to determine the placement of a $field_p$ within the TCAM $segment_i$, where $p = i = 1, 2, 3 \ldots, d$ and d represents the classification dimension, is known as Field Selection Method.

As defined, to address the above problem, information structure of the classification rule databases needs to be analyzed. As per information theory (Han et al. 2011; Hartmann et al. 1982), maximum pruning in selection happens if the information contained in the reference database used for classification is more and vice versa. The rule database structure of popularly used packet classification rule databases (Taylor and Turner 2007) was shown in Table 5. Table 5 infers very interesting structure within the rule databases. The information contained in the source port field and protocol fields was very low, while the information present in the destination port was very high in the majority of the databases.

Also, the properties vary from one class of databases to the other classes. Hence, it is natural to use the field information properties contained in the database as a parameter to the Field Selection Methodology. The algorithm for addressing the problem defined is shown in Figure 5. The algorithm takes the Rule Database S containing N rules as an input and then separates it into individual fields each containing N field values. The unique fields from all the fields are then extracted. Before extracting the same, some fine tuning has to be done to improve the information insight of each field. So, major overlapping fields like default conditions have to be removed. Once the unique field values in each field are determined, the total number of unique values in each field will quantify the information content present in each field. The fields are then sorted into their increasing order of unique values present within them. Then, mapping onto the TCAM segments happens as follows: The field with highest information on the left-most segment (or segment 1, as shown in Figure 6), the next one in the order onto segment 2, and so on. Figure 6 shows the field placement based on the proposed approach for the classification rules given in Table 3.

For the rule database shown in Table 3, the unique values were identified for all the fields by excluding the all-match conditions. Based on the unique fields, the fields were sorted in increasing order of information present within them and the order is $<F3, F4, F1, F2>$. Now, as per the

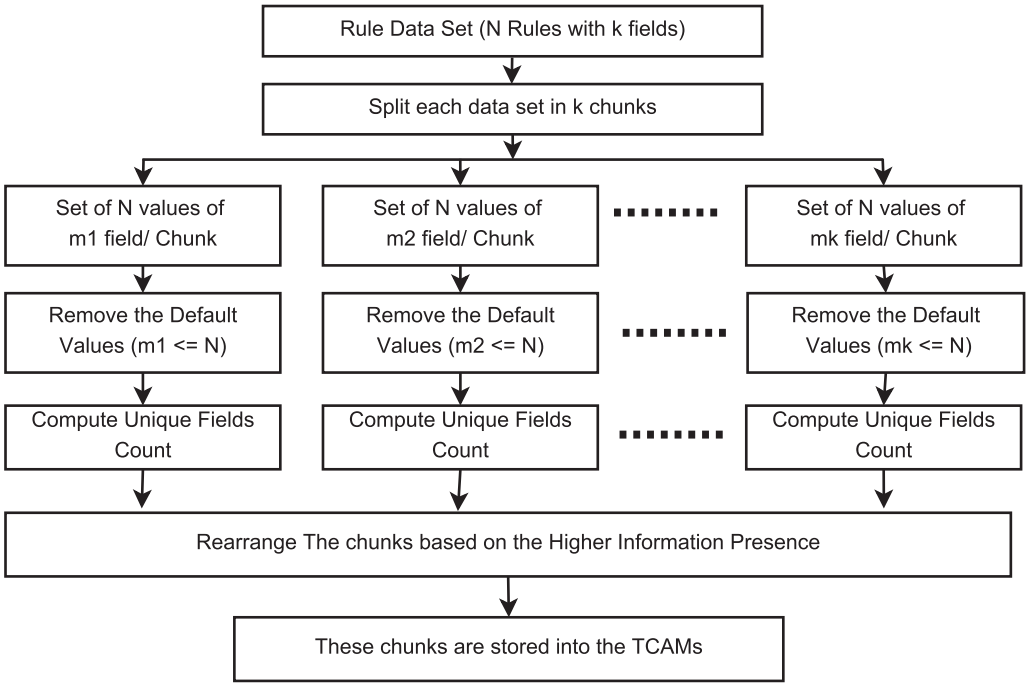


Fig. 5. Algorithm for optimal placement.

proposed placement algorithm, the fields are placed in their appropriate segments like $F3$ in the $segment_1$(the left-most one) and $F2$ in the $segment_4$ (the right-most one).

The same input header as earlier <110, 110, 011, 111> was fed by the NPU to the configurable header distributor. The configurable header distributor will modify the inputs fed to a particular segment in accordance with the new placement order, and hence the modified header input is <011, 111, 110, 110>. In Figure 6, the new set of matchlines that gets activated for each segment and the corresponding rule value were highlighted. The total number of segment matchlines that got activated for the given input trace was 9 as opposed to 17 when regular placement was followed. In Table 4, six flows are considered arbitrarily and the total number of packets in a flow are also shown. The total packets across all the flows numbered 60, and these 60 packets were classified using both the regular segment placing approach and the proposed placement approach. The analysis is shown in Table 6. In the table, the units used for analysis is Matchline Bits (MLB), i.e, the total number of matchlines that gets activated across all the segments and the size of each matchline in terms of bits. The reduction in terms of MLB is significant when compared with both single stride lookup and naive segmented lookup. For few input traces, the matchline activation using Optimal field placement may be high when compared with the naive segmented look-up. One such case is demonstrated in Table 6 for the flow $H6$. However, the cumulative trace outcome can overcome such cases and will lead to optimal matchline activation, as shown in Table 6. The efficiency of the proposed technique has been tested at a large scale against real-world databases, and the results are discussed elaborately in the Section 4.

The above discussion can be summarized as follows: The fields with highest information vary from one rule database to another. This structural property of classification databases is used for modifying the placement order based on the information content present in the fields. This guarantees better purity at each segment, thereby leading to efficient pruning of the search space in the

Table 6. MLB Activation Analysis for Various Look-up Approaches

Flow (Packets/flow)	Single Stride look-up (MLB)	Segmented look-up (MLB)	Segmented look-up with Optimal placement (MLB)
H1 (10)	720	330	210
H2 (15)	1,080	450	315
H3 (5)	360	180	105
H4 (10)	720	180	180
H5 (7)	504	210	168
H6 (13)	936	234	312
Total	4,320	1,584	1,290

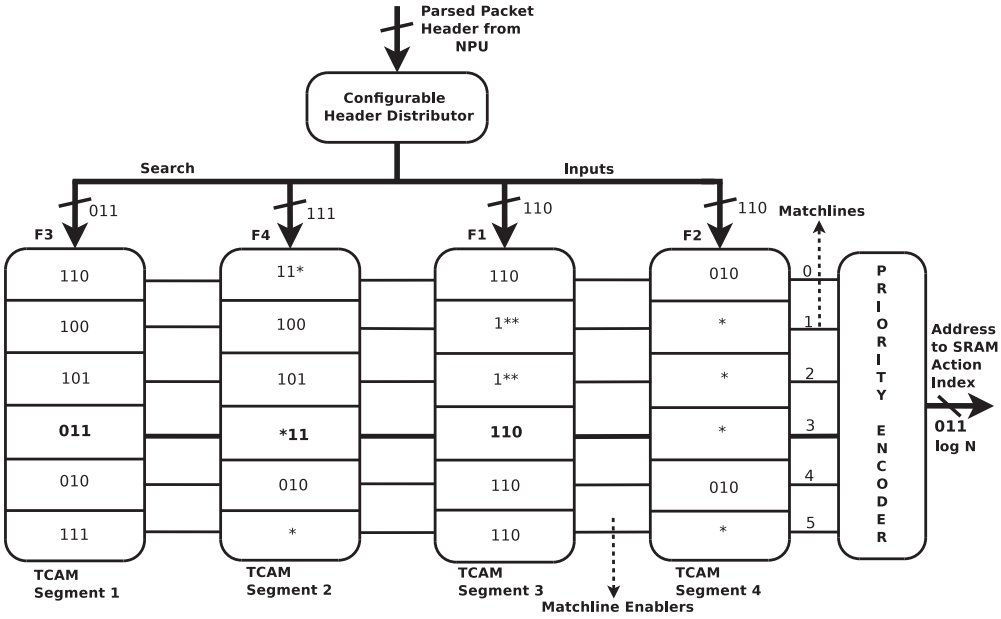


Fig. 6. Matchline segmented TCAM architecture with optimal placement.

subsequent segments. The field information is measured based on the total number of unique field (UF) values present within it. The field with the highest information has to be placed on the left-most chunk segment, and this results in optimal purity at the initial field classification itself. Optimal purity implies a lower number of test cases have to be performed on the remaining fields before yielding the final classification result. Mapping the above to TCAM, placing the field with the highest information on the left-most chunk leads to the optimal purity in the first chunk of the TCAM. Maximum purity results in the minimum number of matchlines being activated for the subsequent chunks. This continues recursively until the last chunk and the final result is associated with the memory location carrying the action index for that particular string. This leads to the significant reduction in the matchline power consumption of the TCAM. As discussed earlier, matchline power contributes to the significant portion of total TCAM power, and hence the total search power consumption will come down. Also, matchline capacitance contributes to the increased TCAM delay, and use of segmented matchline TCAMs will outperform the single stride matchline TCAMs in

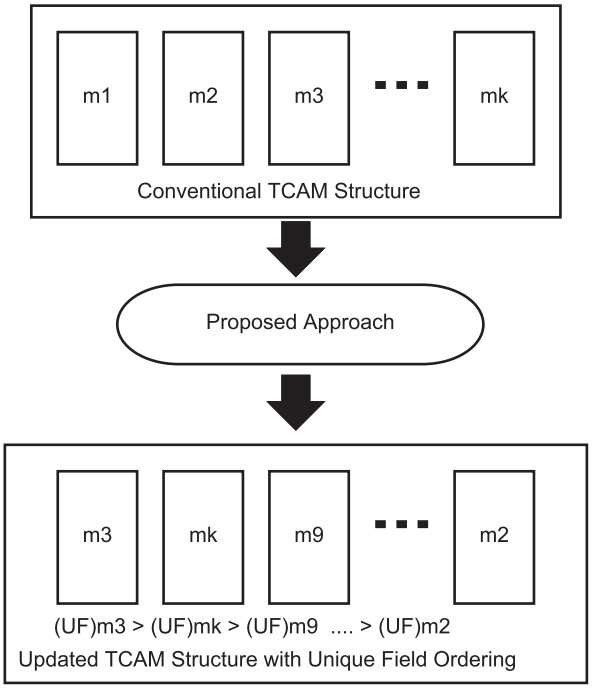


Fig. 7. TCAM structure before and after applying proposed placement ordering.

terms of delay and thereby improve the overall classification throughput. The generalized TCAM field placement structure before and after applying the proposed placement is shown in Figure 7.

3.2 Analysis

3.2.1 Pre-processing. The pre-processing in the proposed approach is required only once at the beginning while the rule set is placed on the TCAM. The pre-processing step involves finding out the information properties of the various fields and sorting them in the order for TCAM placement. Theorem 1 gives the pre-processing complexity for the proposed approach.

Theorem 1. *For a given database of size N, the complexity of finding the unique match conditions in a field, for all the fields in the database is* $O(N^2)$ *if done in a linear way.*

Proof. As explained in Section 3.1, let us consider a database S, across d dimensions containing N rules. The unique match values of each field has to be determined separately. For a field F_1, begin from the item 1 and add it to the list L and increment the count by one. From item 2, compare each item with those present in the list. If the item from the F_1 does not match those in the list, then add it to the list and increment the count every time by 1 or else ignore. The primitive operation here is a simple comparison. The total number of comparisons required for the items scanned in the F_1 from 1 to N are: $0 + 1 + 2 + \cdots\cdots + N - 1$. Asymptotically, the progression reduces to $O(N^2)$. The above scanning has to be done for all the d fields. So, the complexity becomes, $O(dN^2)$. But in real life, the d spans from 5 to 12 (Openflow (McKeown et al. 2008)). This makes $d \ll N$. Hence, the complexity becomes $O(N^2)$ for the pre-processing. Hence, proved. □

The complexity of $O(N^2)$ can be brought down to $O(NlogN)$ if a tree is used for comparison instead of a list. But the pre-processing is required only once at the beginning, and constructing a tree requires additional computation, which makes it inefficient than the linear way.

3.2.2 Look-up Complexity and Commodity TCAM Mapping. The look-up, as explained in Section 3.1, happens in a field-segmented manner. For classifier of d dimensions, the look-up takes d clock cycles, making the look-up complexity $O(d)$. However, using field-level pipelining, it is possible to perform one look-up per clock cycle and hence O(1) can be achieved. The modern-day TCAM memories are mostly available with segmented and pipelined matchlines (Pagiamtzis and Sheikholeslami 2004, 2006) also along with the bit-width programmable (eSilicon 2019). The standard width in the majority of the TCAMs used in switching and Routing is 144 and almost the majority of the works available in the literature (Agrawal and Sherwood 2008; Pagiamtzis and Sheikholeslami 2004, 2006) and the device specifications of the industry follow the following segmentation: (4×34 plus 8—Total $144Bits$). Since each chunk has an adequate number of bits to place any of the IP or Range fields within it, the mapping of changed rule pattern with respect to the information content is straightforward and does not require any major hardware modifications, making the proposed solution compatible with the existing designs.

3.2.3 Incremental Updates. The proposed solution will support incremental updates. The incremental updates refers to one of the three: (1) Inserting, (2) Deleting, or (3) Modifying a rule in the database, while retaining the proposed optimal placement. Modification of rule can be considered as deletion of the rule followed by addition of the modified rule to the database. Performing any of the above three can result in the change of optimal order. Hence, performing the update requires addressing the changes in the optimal placement on the TCAM as well. For the purpose of explanation, only the rule addition is considered here. During pre-processing, a list of elements for every field is constructed along with a count variable for maintaining the list and count of unique field values, respectively. This is done for all the fields in the database, and the lists are preserved in the NPU for assisting the incremental updates. Let us consider a rule R_i came as an update that needs to be added to the classifier. The rule R_i can be written onto the TCAM based on the existing field ordering. Updating a single location in the TCAM takes one standard clock cycle. Once the new rule is added, the classification can be allowed to be done. However, the primary challenge is the change in the optimal order that may happen due to the new rule addition. Since the optimal placement depends on the unique field values, addition/deletion/modification of a rule can affect the existing order. This requires a corresponding change in the existing placement order on the TCAM as well if the current placement order changes at all. In this section, solutions for the incremental updates by considering two TCAM classifier frameworks are inspected: (1) with Tandem TCAMs and (2) with Single TCAM.

Incremental Update with Tandem TCAMs: Commodity switches that are highly sensitive to Quality of Service (QoS) cannot afford to keep the classification on hold until the update is finished. For the same reason, a Tandem TCAM structure (Meiners et al. 2010) is employed by the switches, as shown in Figure 8. The following approach is highly efficient in terms of QoS for the update process. Whenever a new rule comes, update it straightaway onto the Classifier TCAM by making a copy of it in the NPU. Now, the NPU can assess the modifications in the rule database information properties and prepare the new sorted list of the fields. The new rule placement is done on the standby TCAM and it takes N clock cycles for the update. During the update on TCAM 2, the classification will be done by TCAM 1. However, during this period the search space pruning or matchline reduction may not be optimal, as the new placement order has been not updated on the current classifier TCAM. This leads to minor penalty in terms of reduction in the activation of matchlines. However, this is the efficient way to ensure the Quality of Service. Once the update as per the new placement order finishes on the standby TCAM, the NPU can switch the classification onto the updated TCAM and then the optimal reduction can be achieved. The total computational complexity of this update procedure is $O(N)$, as explained in the Theorem 2.

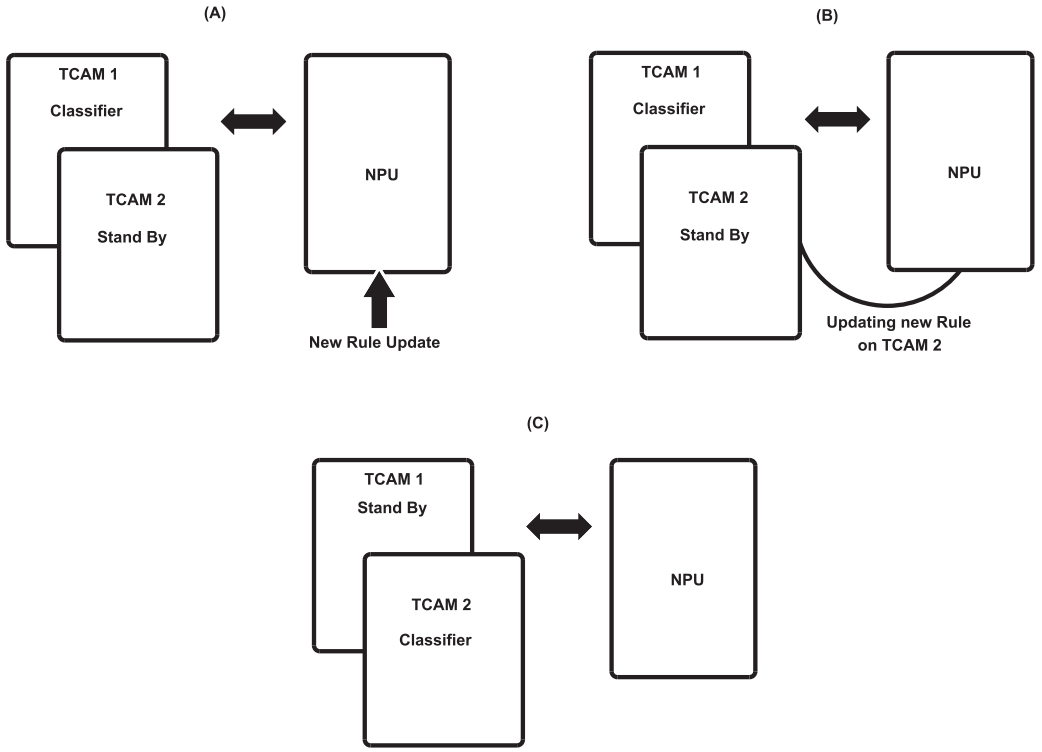


Fig. 8. Incremental update framework along with optimal placement using tandem TCAM structure.

Theorem 2. *The complexity of performing incremental rule update along with maintaining the optimal field placement is* $O(N)$.

Proof. As per the procedure explained above, it takes N comparisons across d dimensions to calculate the modified field placement whenever a new rule is Added/Deleted to the rule database S. Again, $d \ll N$, and hence the complexity of determining the modified field ordering is $O(N)$. Once determined, the N rules have to be updated onto the standby TCAM per the new ordering that takes N clock cycles. So, the total complexity is $T = O(N) + O(N)$, which is $O(N)$. Hence, proved. □

Incremental Update with Single TCAM: The above update procedure works if the router on which the optimal placement is employed comes with two TCAMs operating in tandem. However, in situations where only one TCAM is present, a different procedure is required for performing the update. The update procedure given for the tandem structure addresses both QoS and optimal placement for power efficiency. But when only one TCAM is present for classification and update, a trade-off will exist between both. Hence, an approach called ***Lazy Update*** can be used for providing a better compromise between QoS and optimal placement for power efficiency.

Assume that there are n entries in the TCAM and currently m locations were filled where $m < n$. The remaining $n - m$ locations can be used for the addition of new rules coming as an update. Let us assume a rule R_i came as an update. Lazy Update makes use of the following observation: *It is highly unlikely that a change in the optimal order will happen with addition/deletion/modification of a single rule to the database containing a huge number of rules.* So, when a new rule comes as an update, add it to the TCAM per the current optimal order. Once added, the classification can be

continued while in parallel, the NPU assesses the modifications in the rule database information properties and prepares the new sorted list of the fields. The procedure can be repeated for a batch of rules or for a specific period of time. Later, the rule database can be updated onto the TCAM as per the new optimal order. During the field placement update, the TCAM cannot perform the classification, as only one TCAM is available for both update and classification. Since it is done only periodically or for a batch of updates, it reduces the effect on the Quality of Service to a great extent. Since it is highly unlikely that few rules can actually modify the optimal order, reduction in the search pruning will also be much less. The periodical update complexity that is performed for a batch of rules also remains same as $O(N)$.

A choice of relevant update strategy can be made based on the classifier framework on which the proposed optimal placement is employed. If the classifier comes with two TCAMs operating in tandem, then employing the update strategy given above for the tandem structure effectively addresses both the Quality of Service and power efficiency. Else, lazy update strategy can be employed if only one TCAM is present for performing both update and classification.

4 EVALUATION

In this section, the efficiency of the proposed placement approach in terms of reduction in the search space is evaluated. Here, the more generalized term called **search space** is used to represent the percentage of matchline segments that gets activated. Hence, the term reduction in search space relates to reduction in the total number of activated matchline segments. The corresponding power reduction is evaluated when mapped onto the TCAM using Agrawal and Sherwood (2008). The popularly used standard ClassBench data sets (Taylor and Turner 2007) were used. They consist of three classes of rule databases, namely Access Control Lists (ACL), Firewall (FW), and IP Chaining (IPC) rule sets. Each class contains five databases of various sizes spanning from 100 rules to 10,000 rules. The traces were generated using the trace generator tool given by Taylor and Turner (2007). The trace generator tool takes the rule set against which the traces will be tested, the trace count, and the trace locality as inputs for generating the traces. Generated traces are of scale five to ten, approximately, meaning the number of traces will be five to ten times the total number of rules in the dataset. Also, traces of different locality were combined to ensure that the proposed approach performance is independent of the trace locality. The trace generator offers three types of locality: (1) no locality, (2) low locality, and (3) high locality. Each rule set was tested against two field combinations: one being the naive and another one based on the proposed algorithm. Each rule set was tested against traces generated over five different iterations and the results shown were the average of all the iterations carried out.

4.1 Search Space Reduction

The rule sets and the information content of each field with respect to the number of unique fields within them are shown in Table 5. From Table 5, it can be seen that for *ACL*, Source Port (SP) is a highly impure field. Hence, the classification on *SP* should be performed after performing classification on all other fields. Also, the pure fields and impure fields remained almost the same for various rule sets within the same class, and they varied when the class of rule sets changed. Based on Table 5 rule database properties, the search space reduction, when the packets were classified in the field-segmented manner, is shown in Table 7. The new order for each rule set based on the unique field values is shown in Table 7. For those databases in which the naive order is highly impure, they yield the highest amount of search space reduction when tested against the pure order. The *FW* Class of databases show the highest amount of reduction in terms of search space, with an average of 47.86%. *IPC* class rule databases show the least reduction with an average of 18.27%. Also, it is possible that the original order may remain the same for some

Table 7. Search Space Reduction with Modified Field Ordering

DataBase (Rules Count)	% Reduction in Search Space	Modified Field Ordering
ACL 100 (98)	20.46	(DA, DP, SA, Pr, SP)
ACL1 (752)	13.56	(DA, DP, SA, Pr, SP)
ACL 1K (916)	44.84	(DA, SA, DP, Pr, SP)
ACL 5K (4,415)	17.62	(SA, DA, DP, Pr, SP)
ACL 10K (9,603)	18.78	(SA, DA, DP, Pr, SP)
FW 100 (92)	24.49	(DP, DA, SA, SP, Pr)
FW1 (269)	48.21	(DA, SA, DP, SP, Pr)
FW 1K (791)	54.65	(DA, SA, DP, SP, Pr)
FW 5K (4,653)	54.25	(DA, SA, DP, SP, Pr)
FW 10K (9,311)	**57.73**	(DA, SA, DP, SP, Pr)
IPC 100 (99)	**In Optimal Order**	(SA, DA, SP, DP, Pr)
IPC1 (1,550)	11.95	(SA, DA, DP, SP, Pr)
IPC 1K (938)	**11.11**	(DA, SA, DP, SP, Pr)
IPC 5K (4,460)	32.28	(DA, SA, DP, SP, Pr)
IPC 10K (9,037)	17.77	(DA, SA, DP, SP, Pr)

rule databases, meaning the naive order is also the pure order. In Table 7, *IPC*100 database has retained the naive order, and hence the reduction in the search space will be null. Among the 15 datasets, *ACL*100, which is a synthetic dataset, has shown the search space degradation when compared with the naive order in few of the simulation iterations carried out. This might be due to significant rule overlappings within the fields of the database, whereas the other 14 datasets never exhibited such behavior. To summarize, classifying on the modified order that is obtained based on the information presence in each field will yield better efficiency in terms of reduction in the number of test cases to be performed before arriving at the final classification result. The highest reduction of 57.73% by FW 10K and the lowest reduction of 11.11% by IPC 1K in the search space were achieved by the proposed technique. An average reduction of 30.55% across all 15 databases was observed. Considering a computationally less intensive pre-processing requirement, the reduction in the search space achieved is significant.

4.2 TCAM Search Energy Reduction

The reduction in the search space is mapped to the TCAM as a classifier and thereby evaluated the reduction in the total TCAM energy consumed per access. For evaluation, the TCAM size requirement for each rule database has to be determined. To determine the same, the total number of ternary strings the rule database gets converted into after *range encoding* is required. For the same, the worst-case rule expansion for each database using the three encoding schemes was calculated. Only deterministic range-encoding schemes were considered for evaluation, as they are efficient in terms of supporting quick search and incremental updates. The three deterministic encoding schemes shown in Table 2 were picked for evaluation. The 1-D (1-Dimensional) ranges, 2-D (2-Dimensional Ranges), and Simple Rules statistics were crucial in calculating the total strings for each rule set. Hence, the source port and destination port fields of ClassBench rules were further analyzed and classified each range into either of: (1) Simple Rule, (2) 1-D Rule, or (3) 2-D Rule. The classification was shown in Table 8. For each of the schemes, the total number of strings for each database after range encoding was also shown in Table 8. Once the size is determined, the *energy per search* was determined using the TCAM model (Agrawal and Sherwood 2008). The tool

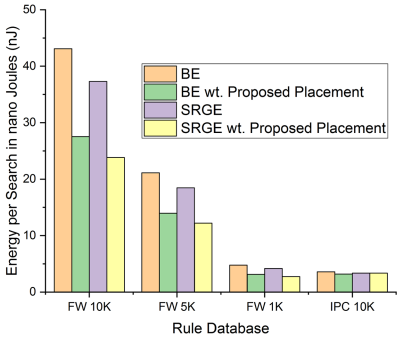
Table 8. Range Properties and Corresponding Expansion of ClassBench Databases

Rule Database	# of 1-D Rules	# of 2-D Rules	# Rules with Worst Case Expansion		
			Binary Encoding	SRGE	Optimal Encoding
ACL 100	7	0	301	287	203
ACL1	165	0	5,537	5,207	3,227
ACL 1K	114	0	4,222	3,994	2,626
ACL 5K	559	0	20,626	19,508	12,800
ACL 10K	1,104	0	41,619	39,411	26,163
FW 100	2	5	4,645	4,061	277
FW1	5	17	15,697	13,715	871
FW 1K	15	67	61,459	53,657	3,093
FW 5K	83	295	272,265	237,879	15,043
FW 10K	165	602	555,294	480,512	30,448
IPC 100	10	0	389	369	249
IPC1	193	1	8,046	7,544	4,476
IPC 1K	97	0	3,751	3,557	2,393
IPC 5K	466	1	18,873	17,825	11,481
IPC 10K	1,006	9	46,302	43,246	24,406

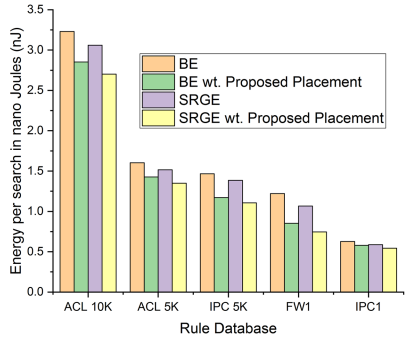
given in Agrawal and Sherwood (2008) takes the top-level parameters: number of rows, bits per row, technology, subbanks, and row divisions as input and provides the TCAM energy per search in nano Joules (nJ) and delay per search in nano seconds (ns) as output. For simulation, we consider a 144-Bit width TCAM with five Sub-Banks for five fields and 45nm technology. The tool provides the total power along with the corresponding Matchline (ML), Searchline (SL), and Priority Encoder (PE) power. The reduction in the search space will only lead to reduction in the total number of matchline segments being activated, and the reduced energy per search is calculated per Equation (1):

$$TE_{Proposed} = (1 - (fraction\ reduction\ in\ search\ space)) \times ML_{Direct} + SL_{Direct} + PE_{Direct} \quad (1)$$

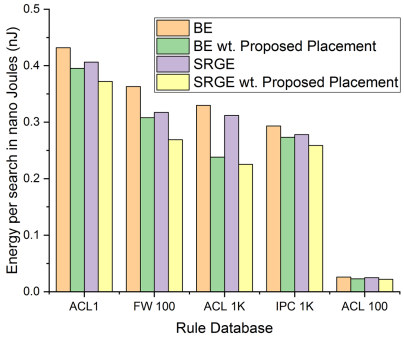
In Equation (1), the term *Direct* in the notation represents the TCAM energy per search obtained when the input parameters were determined for a particular database and fed as an input to the modeling tool. The $TE_{Proposed}$ is the energy per search with respect to the proposed placement search space reduction. The energy reduction achieved for various databases when the proposed solution is integrated with the range-encoding schemes was shown in Figure 9 for Binary Encoding, SRGE (Bremler-Barr and Hendler 2012), and Optimal Encoding (Rottenstreich et al. 2013). The databases were divided among the plots for scaling purposes. An average energy-per-search reduction of 18.85% has been achieved by the proposed placement when tested using the three encoding schemes. A TCAM clocked at $200Mhz$ can perform 200 Million searches per second, and this reduction per search will significantly reduce the total energy consumption of the TCAM coprocessor, thereby reducing the router system power significantly. The delay performance of the segmented TCAM with optimal encoding is shown in Figure 10. The proposed placement approach does not affect the search delay (i.e., it neither increases nor decreases) when compared with the segmented TCAM. Hence, the proposed placement approach has the advantage of not placing any energy-delay trade-off. Figure 10 can be used to determine the total reduction in the search energy over a period of time $\boldsymbol{t}$ for various databases. The impact of the reduction over a time period of $1s$ is shown in Figure 11 for two databases. The databases chosen were IPC 1K (least reduction in



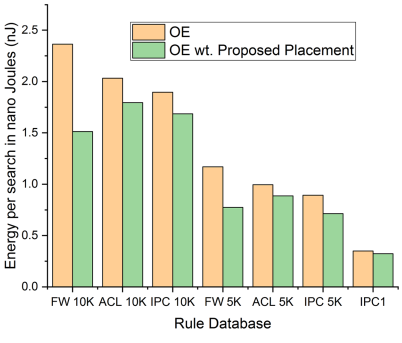
(a)



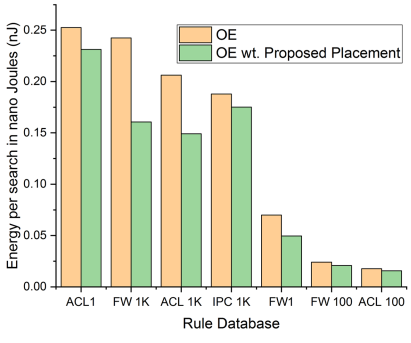
(b)



(c)



(d)



(e)

Fig. 9. Energy-per-search comparison w.r.t Binary (BE) and SRG Encoding (SRGE) using proposed placement, shown in Figures (a), (b), and (c); and for Optimal Encoding (OE), shown in Figures (d) and (e).

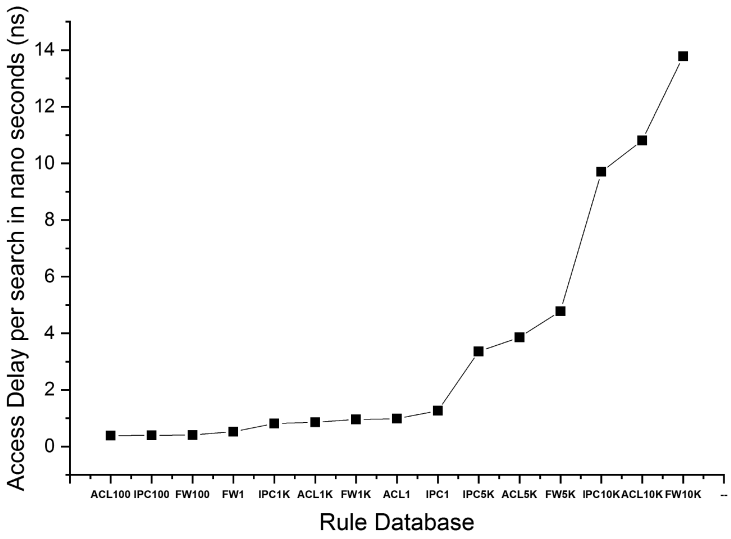


Fig. 10. Delay per search for segmented TCAM with optimal encoding.

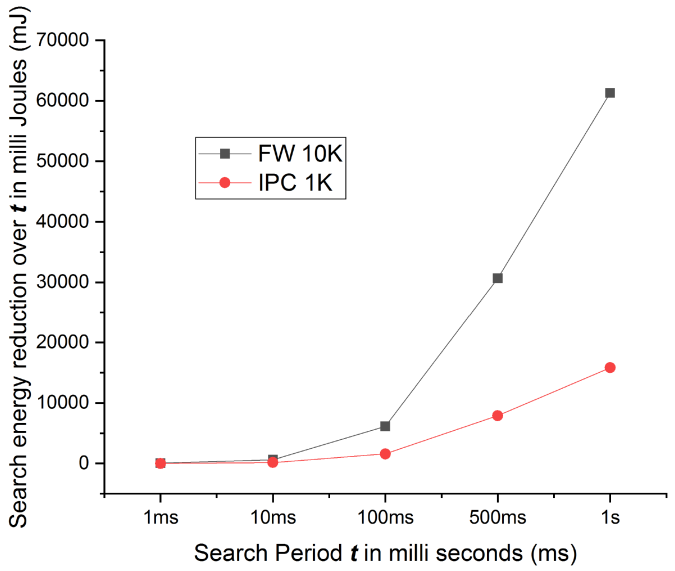


Fig. 11. Search energy reduction over a period of time.

the search space) and FW 10K (highest reduction in the search space) for representation. Although the reduction in the search space is less for IPC 1K, the reduction achieved over a period of 1s is 15.82J, which is significantly high. Similarly, for FW 10K, the reduction is 61.96J, as shown in Figure 11. It demonstrates the impact of the proposed optimal placement on the search-engine energy consumption for the routers deployed in real time. For example, the commercial TCAMs provided by eSilicon (2019) and RENESAS (2015) comes with segmented matchlines/search word widths for achieving the search power reduction. In such TCAMs, the use of optimal field placement for

placing the appropriate fields on the appropriate segments can further bring down their search energy consumption.

5 CONCLUSION

In this work, an efficient rule field placement algorithm for the TCAM-based packet classification is proposed. The order of placement is based on the field information property, which is proportional to the total number of unique fields present in a particular field of the rule database. When compared with the traditional placement order, the proposed placement will result in the least number of matchline segments getting activated within a segmented TCAM. The solution can be easily mapped to the existing commodity TCAMs with computationally less-intensive pre-processing requirements. The solution supports incremental updates with linear complexity and uninterrupted classification. An average reduction of 30.55% in search space is attained when tested on ClassBench databases. This reduction in the search space when mapped to the TCAM engine resulted in approximately 19% reduction in per-search energy of the TCAM when tested in line with various state-of-the-art range-encoding solutions. The proposed rule-mapping solution can significantly bring down the energy consumption of TCAM-based network switches and routers while leaving the search performance unaffected. This can offer good flexibility to designers in terms of addressing the energy-consumption barriers, thereby reducing the system cost.

REFERENCES

Banit Agrawal and Timothy Sherwood. 2008. Ternary CAM power and delay model: Extensions and uses. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 16, 5 (2008), 554–564.

Florin Baboescu and George Varghese. 2001. Scalable packet classification. *ACM SIGCOMM Comput. Commun. Rev.* 31, 4 (2001), 199–210.

Tania Banerjee, Sartaj Sahni, and Gunasekaran Seetharaman. 2015. PC-TRIO: A power efficient TCAM architecture for packet classifiers. *IEEE Trans. Comput.* 64, 4 (2015), 1104–1118.

Anat Bremler-Barr and Danny Hendler. 2012. Space-efficient TCAM-based classification using gray coding. *IEEE Trans. Comput.* 61, 1 (2012), 18–30.

eSilicon Corporation. 2019. Retrieved from: https://www.esilicon.com/products/high-performance-networking-computing-ip/tcam-and-bcam-compilers/.

Miad Faezipour and Mehrdad Nourani. 2009. Wire-speed TCAM-based architectures for multimatch packet classification. *IEEE Trans. Comput.* 58, 1 (2009), 5–17.

Pankaj Gupta and Nick McKeown. 1999. Packet classification on multiple fields. *ACM SIGCOMM Comput. Commun. Rev.* 29, 4 (1999), 147–160.

Pankaj Gupta and Nick McKeown. 2000. Classifying packets with hierarchical intelligent cuttings. *IEEE Micro* 20, 1 (2000), 34–41.

Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data Mining: Concepts and Techniques*. Elsevier.

C. Hartmann, Pramod Varshney, Kishan Mehrotra, and C. Gerberich. 1982. Application of information theory to the construction of efficient decision trees. *IEEE Trans. Inform. Theory* 28, 4 (1982), 565–577.

Weirong Jiang and Viktor K. Prasanna. 2009. Field-split parallel architecture for high performance multi-match packet classification using FPGAs. In *Proceedings of the 21st Symposium on Parallelism in Algorithms and Architectures*. ACM, 188–196.

Weirong Jiang and Viktor K. Prasanna. 2012. Scalable packet classification on FPGA. *IEEE Trans. Very Large Scale Integ. (VLSI) Systems* 20, 9 (2012), 1668–1680.

Xianfeng Li, Yuanxin Lin, and Wenjun Li. 2016. GreenTCAM: A memory-and energy-efficient TCAM-based packet classification. In *Proceedings of the International Conference on Computing, Networking and Communications (ICNC'16)*. IEEE, 1–6.

Alex X. Liu, Chad R. Meiners, and Eric Torng. 2010. TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs. *IEEE/ACM Transactions on Networking (TON)* 18, 2 (2010), 490–500.

Alex X. Liu, Chad R. Meiners, and Eric Torng. 2016. Packet classification using binary content addressable memory. *IEEE/ACM Transactions on Networking* 24, 3 (2016), 1295–1307.

Alex X. Liu, Chad R. Meiners, and Yun Zhou. 2008. All-match based complete redundancy removal for packet classifiers in TCAMs. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM'08)*. IEEE, 111–115.

Huan Liu. 2002. Efficient mapping of range classifier into ternary-CAM. In *Proceedings of the 10th Symposium on High Performance Interconnects*. IEEE, 95–100.

Yadi Ma and Suman Banerjee. 2012. A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. ACM, 335–346.

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* 38, 2 (2008), 69–74.

Chad R. Meiners, Alex X. Liu, and Eric Torng. 2010. *Hardware Based Packet Classification for High Speed Internet Routers*. Springer Science & Business Media.

Chad R. Meiners, Alex X. Liu, and Eric Torng. 2011a. Topological transformation approaches to TCAM-based packet classification. *IEEE/ACM Trans. Netw.* 19, 1 (2011), 237–250.

Chad R. Meiners, Alex X. Liu, Eric Torng, and Jignesh Patel. 2011b. Split: Optimizing space, power, and throughput for TCAM-based classification. In *Proceedings of the ACM/IEEE 7th Symposium on Architectures for Networking and Communications Systems*. IEEE Computer Society, 200–210.

Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. 2009. *CACTI 6.0: A Tool to Model Large Caches*. Technical Report HPL-2009-85. *HP Laboratories* (2009), 22–31.

Shanmugakumar Murugesan and Noor Mahammad Sk. 2017. A novel range matching architecture for packet classification without rule expansion. *ACM Trans. Design Auto. Electron. Syst.* 23, 1 (2017), 8.

Kostas Pagiamtzis and Ali Sheikholeslami. 2004. A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme. *IEEE J. Solid-State Circ.* 39, 9 (2004), 1512–1519.

Kostas Pagiamtzis and Ali Sheikholeslami. 2006. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE J. Solid-state Circ.* 41, 3 (2006), 712–727.

Ioannis Papaefstathiou and Vassilis Papaefstathiou. 2007. Memory-efficient 5D packet classification at 40 Gbps. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*. IEEE, 1370–1378.

Yaxuan Qi, Lianghong Xu, Baohua Yang, Yibo Xue, and Jun Li. 2009. Packet classification algorithms: From theory to practice. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM'09)*. IEEE, 648–656.

Renesas Electronics Corporation. 2015. Retrieved from: https://www.renesas.com/jp/ja/doc/products/memory/r10cp0001eu0100_tcam.pdf.

Ori Rottenstreich, Rami Cohen, Danny Raz, and Isaac Keslassy. 2013. Exact worst case TCAM rule expansion. *IEEE Trans. Comput.* 62, 6 (2013), 1127–1140.

Ori Rottenstreich, Isaac Keslassy, Avinatan Hassidim, Haim Kaplan, and Ely Porat. 2016. Optimal in/out TCAM encodings of ranges. *IEEE/ACM Trans. Netw.* 24, 1 (2016), 555–568.

Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. 2003. Packet classification using multidimensional cutting. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 213–224.

Haoyu Song and John W. Lockwood. 2005. Efficient packet classification for network intrusion detection using FPGA. In *Proceedings of the ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays*. ACM, 238–245.

Ed Spitznagel, David E. Taylor, and Jonathan S. Turner. 2003. Packet classification using extended TCAMs. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, Vol. 3. 120–131.

David E. Taylor. 2005. Survey and taxonomy of packet classification techniques. *ACM Comput. Surv.* 37, 3 (2005), 238–275.

David E. Taylor and Jonathan S. Turner. 2005. Scalable packet classification using distributed crossproducing of field labels. In *Proceedings of the 24th Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1. IEEE, 269–280.

David E. Taylor and Jonathan S. Turner. 2007. Classbench: A packet classification benchmark. *IEEE/ACM Trans. Netw.* 15, 3 (2007), 499–511.

Jan Van Lunteren and Ton Engbersen. 2003. Fast and scalable packet classification. *IEEE J. Select. Areas. Commun.* 21, 4 (2003), 560–571.

Matteo Varvello, Rafael Laufer, Feixiong Zhang, and T. V. Lakshman. 2016. Multilayer packet classification with graphics processing units. *IEEE/ACM Trans. Netw.* 24, 5 (2016), 2728–2741.

Rihua Wei, Yang Xu, and H. Jonathan Chao. 2016. Finding nonequivalent classifiers in Boolean space to reduce TCAM usage. *IEEE/ACM Trans. Netw.* 24, 2 (2016), 968–981.

Bo Xu, Dongyi Jiang, and Jun Li. 2005. HSM: A fast packet classification algorithm. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Vol. 1. IEEE, 987–992.

Yang Xu, Zhaobo Liu, Zhuoyuan Zhang, and H. Jonathan Chao. 2014. High-throughput and memory-efficient multimatch packet classification based on distributed and pipelined hash tables. *IEEE/ACM Trans. Netw.* 22, 3 (2014), 982–995.

Kai Zheng, Hao Che, Zhijun Wang, Bin Liu, and Xin Zhang. 2006. DPPC-RE: TCAM-based distributed parallel packet classification with range encoding. *IEEE Trans. Comput.* 55, 8 (2006), 947–961.

Received November 2018; revised February 2019; accepted April 2019