TCAM Razor: A Systematic Approach Towards Minimizing Packet Classifiers in TCAMs

Chad R. Meiners Alex X. Liu Eric Torng
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, U.S.A.
{meinersc, alexliu, torng}@cse.msu.edu

Abstract— **Packet classification is the core mechanism that enables many networking services on the Internet such as firewall packet filtering and traffic accounting. Using Ternary Content Addressable Memories (TCAMs) to perform high-speed packet classification has become the de facto standard in industry. TCAMs classify packets in constant time by comparing a packet with all classification rules of ternary encoding in parallel.**

Despite their high speed, TCAMs suffer from the well-known range expansion problem. As packet classification rules usually have fields specified as ranges, converting such rules to TCAM-compatible rules may result in an explosive increase in the number of rules. This is not a problem if TCAMs have large capacities. Unfortunately, TCAMs have very limited capacity, and more rules means more power consumption and more heat generation for TCAMs. Even worse, the number of rules in packet classifiers has been increasing rapidly with the growing number of services deployed on the Internet.

To address the range expansion problem of TCAMs, we consider the following problem: given a packet classifier, how can we generate another semantically equivalent packet classifier that requires the least number of TCAM entries? In this paper, we propose a systematic approach, the *TCAM Razor*, that is effective, efficient, and practical. In terms of effectiveness, our TCAM Razor prototype achieves a total compression ratio of 3.9%, which is significantly better than the previously published best result of 54%. In terms of efficiency, our TCAM Razor prototype runs in seconds, even for large packet classifiers. Finally, in terms of practicality, our TCAM Razor approach can be easily deployed as it does not require any modification to existing packet classification systems, unlike many previous range expansion solutions.

I. Introduction

Packet classification, which has been widely deployed on the Internet, is the core mechanism that enables routers to perform many networking services such as firewall packet filtering, virtual private networks (VPNs), network address translation (NAT), quality of service (QoS), load balancing, traffic accounting and monitoring, differentiated services (Diffserv), etc. As more services are deployed on the Internet, packet classification grows in demand and importance.

The function of a packet classification system is to map each packet to a decision (*i.e.*, action) according to a sequence (*i.e.*, ordered list) of rules, which is called a packet classifier. Each rule in a packet classifier has a predicate over some packet header fields and a decision to be performed upon the packets that match the predicate. To resolve possible conflicts among rules in a classifier, the decision for each packet is the decision of the first (i.e., highest priority) rule that the packet matches. Table I shows an example packet classifier of two rules. The format of these rules is based upon the format used in Access Control Lists on Cisco routers.

A. Motivation

There are two types of packet classification schemes: software-based and hardware-based. Many software-based packet classification algorithms and techniques have been proposed in the past decade (*e.g.*, [4], [5], [8], [10], [13], [19], [20], [22], [26], [27]). Based on complexity bounds from computational geometry [18], for packet classification with n rules and $d > 3$ fields, the "best" software-based packet classification algorithms use either $O(n^d)$ space and $O(\log n)$ time or $O(n)$ space and $O(\log^{d-1} n)$ time. Many software-based solutions are either too slow (such as linear search) or too memory intensive (such as RFC [10]). Decision-tree based packet classification algorithms, which were pioneered by Woo [27] and Gupta and McKeown [11], seem to achieve better time-space tradeoffs. However, they may not work as well in the future as they have exploited statistical characteristics of packets classifiers to achieve the above time-space tradeoffs, and it has been observed that these statistical characteristics are changing [14].

Due to the inherent limitations of software-based packet classification algorithms, more and more packet classification systems are hardware-based; specifically, most packet classification systems now use Ternary Content Addressable Memories (TCAMs). A TCAM is a memory chip where each entry can store a packet classification rule that is encoded in ternary format. Given a packet, the TCAM hardware can compare the packet with all stored rules in parallel and then return the decision of the first rule that the packet matches. Thus, it takes $O(1)$ time to find the decision for any given packet. Because of their high speed, TCAMs have become the de facto industrial standard for high speed packet classification [1], [14]. In 2003, most packet classification devices shipped were TCAM-based [2]. More than 6 million TCAM devices were deployed worldwide in 2004 [2].

Despite their high speed, TCAMs have their own limitations with respect to packet classification.

Rule	Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
r_1	1.2.3.0/24	192.168.0.1	[1,65534]	[1,65534]	TCP	accept
r_2	*	*	*	*	*	discard

TABLE I

AN EXAMPLE PACKET CLASSIFIER

Rule	Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
r_1	1.2.3.0/24	192.168.0.1	0	*	*	discard
r_2	1.2.3.0/24	192.168.0.1	65535	*	*	discard
r_3	1.2.3.0/24	192.168.0.1	*	0	*	discard
r_4	1.2.3.0/24	192.168.0.1	*	65535	*	discard
r_5	1.2.3.0/24	192.168.0.1	[0,65535]	[0,65535]	TCP	accept
r_6	*	*	*	*	*	discard

TABLE II

TCAM RAZOR OUTPUT FOR THE EXAMPLE PACKET CLASSIFIER IN FIGURE I

a) Range expansion: TCAMs can only store rules that are encoded in ternary format. In a typical packet classification rule, source IP address, destination IP address, and protocol type are specified in prefix format, which can be directly stored in TCAMs, but source and destination port numbers are specified in ranges (*i.e.*, integer intervals), which need to be converted to one or more prefixes before being stored in TCAMs. This can lead to a significant increase in the number of TCAM entries needed to encode a rule. For example, 30 prefixes are needed to represent the single range $[1, 65534]$, so $30 \times 30 = 900$ TCAM entries are required to represent the single rule r_1 in Table I.

b) Low capacity: TCAMs have limited capacity. The largest TCAM chip available on the market has 18Mb while 2Mb and 1Mb chips are most popular [2]. Given that each TCAM entry has 144 bits and a packet classification rule may have a worst expansion factor of 900, it is possible that an 18Mb TCAM chip cannot store all the required entries for a modest packet classifier of only 139 rules. While the worst case may not happen in reality, this is certainly an alarming issue. Furthermore, TCAM capacity is not expected to increase dramatically in the near future due to other limitations that we will discuss next.

c) High power consumption and heat generation: TCAM chips consume large amounts of power and generate large amounts of heat. For example, a 1Mb TCAM chip consumes 15-30 watts of power. Power consumption together with the consequent heat generation is a serious problem for core routers and other networking devices.

d) Large board space occupation: TCAMs occupy much more board space than SRAMs. For networking devices such as routers, area efficiency of the circuit board is a critical issue.

e) High hardware cost: TCAMs are expensive. For example, a 1Mb TCAM chip costs about $200 \sim 250$ U.S. dollars. TCAM cost is a significant fraction of router cost.

B. The Problem

In this paper, we consider the following TCAM Minimization Problem: *given a packet classifier, how can we generate another semantically equivalent packet classifier that requires the least number of TCAM entries?* Two packet classifiers are (semantically) equivalent if and only if they have the same decision for every packet. For example, the two packets classifiers in Tables I and II are equivalent; however, the one in Table I requires 900 TCAM entries, and the one in Table II requires only 6 TCAM entries.

Solving this problem helps to address the limitations of TCAMs. As we reduce the number of TCAM entries required, we can use smaller TCAMs, which results in less board space and lower hardware cost. Furthermore, reducing the number of rules in a TCAM directly reduces power consumption and heat generation because the energy consumed by a TCAM grows linearly with the number of ternary rules it stores [28].

C. Our Solution: TCAM Razor

While the optimal solution to the above problem is conceivably NP-hard, in this paper, we propose a practical algorithmic solution using three techniques: decision diagrams, dynamic programming, and redundancy removal. Our solution consists of the following four basic steps. First, convert a given packet classifier to a reduced decision diagram, which is the canonical representation of the semantics of the given packet classifier. Second, for every nonterminal node in the decision diagram, minimize the number of prefixes associated with its outgoing edges using dynamic programming. Third, generate rules from the decision diagram. Last, remove redundant rules. As an example, running our algorithms on the packet classifier in Table I will yield the one in Table II.

Our solution is effective, efficient, and practical. In terms of effectiveness, our approach achieves a total compression ratio of 3.9% on real-life packet classifiers, which is significantly better than the previously published best result of 54% [6]. In terms of efficiency, our approach runs in seconds, even for large packet classifiers. Finally, in terms of practicality, our approach can be easily deployed as it does not require any modification of existing packet classification systems. In comparison, a number of previous solutions require hardware and architecture modifications to existing packet classification systems, making their adoption by networking manufacturers and ISPs much less likely.

We name our solution "TCAM Razor" following the principle of Occam's razor: *"Of two equivalent theories or explanations, all other things being equal, the simpler one is to be preferred."* In our context, of all packet classifiers that are equivalent, the one with the least number of TCAM entries is preferred.

The rest of this paper proceeds as follows. We start by reviewing related work in Section II. In Section III, we formally define the TCAM minimization problem and related terms. In Section IV, we discuss the weighted one-dimensional TCAM minimization problem. In Section V, we give a solution to the multi-dimensional TCAM minimization problem. In Section VI, we show the experimental results on both real-life and synthetic packet classifiers. Finally, we give concluding remarks in Section VII.

II. Related Work

Many software solutions have been proposed for finding the decision of the first rule that a packet matches in a given packet classifier (*e.g.*, [4], [5], [8], [10], [13], [19], [20], [22], [26], [27]). A comprehensive survey of this work is given in [24].

Recently, hardware packet classification systems based on TCAMs have been widely deployed due to their $O(1)$ classification time. This has led to a significant amount of work that explores ways to cope with the well-known range expansion problem. These solutions fall into three broad categories: (1) *TCAM modification*, which requires changing TCAM hardware circuits, (2) *range encoding*, which does not require changing TCAM hardware circuits, but does require preprocessing for every packet, and (3) *classifier minimization*, which does not require changing TCAM hardware circuits nor preprocessing for any packet.

TCAM Modification: The basic idea is to modify TCAM circuits for packet classification purposes. For example, Spitznagel *et al.* proposed adding comparators at each entry level to better accommodate range matching [21]. This is an important research direction. However, any solutions from this research line will not be deployed for many years due to issues of cost and development [14]. Furthermore, changing the ternary nature of TCAMs makes such TCAMs less generally applicable to applications other than packet classification.

Range Encoding: The basic idea is to re-encode ranges that appear in a packet classifier and then store the re-encoded rules in the TCAM. When a packet comes, the packet needs to be preprocessed according to the re-encoding scheme such that the packet, after preprocessing, can be used as a search key for the TCAM. Several range encoding schemes have been proposed [14], [17], [25]. While the TCAM circuit does not need to be modified to implement range encoding, the system hardware does need to be reconfigured to allow for preprocessing of packets, and the delay caused by packet preprocessing could be problematic.

Classifier Minimization: The basic idea is to convert a given packet classifier to another semantically equivalent packet classifier that requires fewer TCAM entries. These solutions are the most likely to be deployed by networking vendors and ISPs because they require no changes to TCAM hardware or existing packet classification systems and incur no preprocessing overhead for packets. Our work, along with [3], [6], [7], [16], [23], falls into this category.

Three papers focus on one-dimensional and two dimensional packet classifiers. Draves *et al.* proposed an optimal solution for one-dimensional packet classifiers in the context of minimizing routing tables in [7]. Subsequently, in the same context of minimizing routing tables, Suri *et al.* proposed an optimal dynamic programming solution for one-dimensional packet classifiers. They also observed that a generalization of the dynamic program was optimal for two-dimensional packet classifiers in which two rules either are non-overlapping or one contains the other geometrically [23]. Suri *et al.* noted that their dynamic program would not be optimal for packet classifiers with more than 2 dimensions. In our studies, we have extended and implemented Suri *et al.*'s algorithm to minimize 5-dimensional packet classifiers. Unfortunately, the extended algorithm is prohibitively slow even for a packet classifier with just a few rules. Recently, Applegate *et al.* proposed an optimal solution for packet classifiers with two dimensions in which each rule must have one field specified as the whole domain of the field and there are only 2 decisions [3].

Only two papers have considered minimizing packet classifiers with more than 2 dimensions. In [16], Liu and Gouda proposed the first algorithm to remove all the redundant rules in a packet classifier, which consequently reduces the number of TCAM entries needed. In [6], Dong *et al.* observed that both expanding and trimming ranges so that they correspond to prefixes can result in fewer TCAM entries. Our TCAM Razor handles these special cases and more. As we demonstrate in Section VI, TCAM Razor significantly outperforms Liu and Gouda's redundancy removal technique and Dong *et al.*'s heuristics. For example, the total compression ratios for TCAM Razor, redundancy removal, and Dong *et al.*'s scheme are 3.9%, 35%, and 54% respectively. Furthermore, the running time of Dong *et al.*'s techniques are not reported. In comparison, TCAM Razor runs in seconds on a mediocre desktop PC, even for large packet classifiers.

It is not surprising that TCAM Razor outperforms the heuristics of Dong *et al.*. First, although TCAM Razor and Dong *et al.*'s heuristics both process packet classifiers one dimension at a time, TCAM Razor is guaranteed to achieve optimal compression on that dimension, but Dong *et al.*'s heuristics are not. Specifically, TCAM Razor handles all the special cases that Dong *et al.* identify in a systematic fashion. Second, while packet classifier semantics are highly dependent on rule order given their first-match semantics, TCAM Razor reduces the influence of rule order by converting the given packet classifier to a reduced decision diagram, which is a canonical representation of the given packet classifier. On the other hand, Dong *et al.* process rules in their original order, looking at one rule at a time for optimization possibilities.

III. Formal Definitions

We now formally define the concepts of fields, packets, packet classifiers, and the TCAM Minimization Problem. A *field* F_i is a variable of finite length (*i.e.*, of a finite number of bits). The domain of field F_i of w bits, denoted $D(F_i)$, is $[0, 2^w - 1]$. A *packet* over the d fields $F_1, \cdots, F_d$ is a d-tuple $(p_1, \cdots, p_d)$ where each p_i $(1 \leq i \leq d)$ is an element of $D(F_i)$. Packet classifiers usually check the following five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. The length of these packet fields are 32, 32, 16, 16, and 8, respectively. We use Σ to denote the set of all packets over fields $F_1, \cdots, F_d$. It follows that Σ is a finite set and $|\Sigma| = |D(F_1)| \times \cdots \times |D(F_d)|$, where $|\Sigma|$ denotes the number of elements in set Σ and $|D(F_i)|$ denotes the number of elements in set $D(F_i)$.

A *rule* has the form $\langle predicate \rangle \rightarrow \langle decision \rangle$. A $\langle predicate \rangle$ defines a set of packets over the fields F_1 through F_d, and is specified as $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$ where each S_i is a subset of $D(F_i)$ and is specified as either a prefix or a range. A *prefix* $\{0,1\}^k\{*\}^{w-k}$ with k leading 0s or 1s for a packet field of length w denotes the range $[\{0,1\}^k\{0\}^{w-k}, \{0,1\}^k\{1\}^{w-k}]$. For example, prefix 01** denotes the range $[0100, 0111]$. A rule $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d \rightarrow \langle decision \rangle$ is a *prefix rule* if and only if each S_i is represented as a prefix.

When using a TCAM to implement a packet classifier, we typically require that all rules be prefix rules. However, in a typical packet classifier rule, some fields such as source and destination port numbers are represented as ranges rather than prefixes. This leads to *range expansion*, the process of converting a rule that may have fields represented as ranges into one or more prefix rules. In range expansion, each field of a rule is first expanded separately. The goal is to find a minimum set of prefixes such that the union of the prefixes corresponds to the range. For example, if one 3-bit field of a rule is the range $[1, 6]$, a corresponding minimum set of prefixes would be 001, 01$*$, 10$*$, 110. The worst-case range expansion of a $w-$bit range results in a set containing $2w - 2$ prefixes [12]. The next step is to compute the cross product of each set of prefixes for each field, resulting in a potentially large number of prefix rules. In Section I, the range expansion of rule r_1 in Table I resulted in $30 \times 30 = 900$ prefix rules.

A packet $(p_1, \cdots, p_d)$ *matches* a predicate $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$ and the corresponding rule if and only if the condition $p_1 \in S_1 \wedge \cdots \wedge p_d \in S_d$ holds. We use α to denote the set of possible values that $\langle decision \rangle$ can be. For firewalls, typical elements of α include accept, discard, accept with logging, and discard with logging.

A sequence of rules $\langle r_1, \cdots, r_n \rangle$ is *complete* if and only if for any packet p, there is at least one rule in the sequence that p matches. To ensure that a sequence of rules is complete and thus is a packet classifier, the predicate of the last rule is usually specified as $F_1 \in D(F_1) \wedge \cdots F_d \in \wedge D(F_d)$. A *packet classifier* f is a sequence of rules that is complete. The size of f, denoted $|f|$, is the number of rules in f. A packet classifier f is a *prefix packet classifier* if and only if every rule in f is a prefix rule.

Two rules in a packet classifier may overlap; that is, there exists at least one packet that matches both rules. Furthermore, two rules in a packet classifier may conflict; that is, the two rules not only overlap but also have different decisions. Packet classifiers typically resolve conflicts by employing a first-match resolution strategy where the decision for a packet p is the decision of the first (i.e., highest priority) rule that p matches in f. The decision that packet classifier f makes for packet p is denoted $f(p)$.

We can think of a packet classifier f as defining a many-to-one mapping function from Σ to α, where Σ denotes the set of all possible packets and α denotes the set of all possible decisions. Two packet classifiers f_1 and f_2 are *equivalent*, denoted $f_1 \equiv f_2$, if and only if they define the same mapping function from Σ to α; that is, for any packet $p \in \Sigma$, we have $f_1(p) = f_2(p)$. For any packet classifier f, we use $\{f\}$ to denote the set of packet classifiers that are equivalent to f. Now we are ready to define the *TCAM Minimization Problem*.

*Definition 3.1 (**TCAM Minimization Problem**):* Given a packet classifier f_1, find a prefix packet classifier $f_2 \in \{f_1\}$ such that for any prefix packet classifier $f \in \{f_1\}$, the condition $|f_2| \leq |f|$ holds.

IV. One-dimensional TCAM Minimization

We first consider the special problem of *weighted* one-dimensional TCAM minimization, whose solution is used in the next section as a building block for multi-dimensional TCAM minimization. Given a one-dimensional packet classifier f of n prefix rules $\langle r_1, r_2, \cdots, r_n \rangle$, where $\{Decision(r_1), Decision(r_2), \cdots, Decision(r_n)\} = \{d_1, d_2, \cdots, d_z\}$ and each decision d_i is associated with a cost $Cost(d_i)$ (for $1 \leq i \leq z$), we define the cost of packet classifier f as follows:

$$Cost(f) = \sum_{i=1}^{n} Cost(Decision(r_i))$$

Based upon the above definition, the problem of weighted one-dimensional TCAM minimization is stated as follows.

Definition 4.1: (**Weighted One-dimensional TCAM Minimization Problem**) Given a one-dimensional packet classifier f_1 where each decision is associated with a cost, find a prefix packet classifier $f_2 \in \{f_1\}$ such that for any prefix packet classifier $f \in \{f_1\}$, the condition $Cost(f_2) \leq Cost(f)$ holds. $\square$

The problem of one-dimensional TCAM minimization (with uniform cost) has been studied in [7], [23] in the context of compressing routing tables. In this paper, we extend the dynamic programming solution in [23] to solve the weighted one-dimensional TCAM minimization. There are three key observations:

1) For any one-dimensional packet classifier f on $\{*\}^w$, we can always change the predicate of the last rule to

be $\{*\}^w$ without changing the semantics of the packet classifier. This follows from the completeness property of packet classifiers.

2) Consider any one-dimensional packet classifier f on $\{*\}^w$. Let f' be f appended with rule $\{*\}^w \rightarrow d$, where d can be any decision. The observation is that $f \equiv f'$. This is because the new rule is redundant in f' since f must be complete. A rule in a packet classifier is *redundant* if and only if removing the rule from the packet classifier does not change the semantics of the packet classifier.

3) Any prefix $\{0,1\}^k\{*\}^{w-k}$ $(0 \leq k \leq w)$ satisfies one of the following three mutually exclusive conditions:
   a) $\{0,1\}^k\{*\}^{w-k} \subseteq 0\{*\}^{w-1}$,
   b) $\{0,1\}^k\{*\}^{w-k} \subseteq 1\{*\}^{w-1}$,
   c) $\{0,1\}^k\{*\}^{w-k} = \{*\}^w$.

   This property allows us to divide a problem on $\{0,1\}^k\{*\}^{w-k}$ into two sub-problems: one on $\{0,1\}^k0\{*\}^{w-k-1}$, and the other one on $\{0,1\}^k1\{*\}^{w-k-1}$. This divide-and-conquer strategy can be applied recursively.

Based on the above three observations, we formulate an optimal dynamic programming solution to the weighted one-dimensional TCAM minimization problem.

Let $\mathcal{P}$ denote a prefix $\{0,1\}^k\{*\}^{w-k}$. We use $\underline{\mathcal{P}}$ to denote the prefix $\{0,1\}^k0\{*\}^{w-k-1}$, and $\overline{\mathcal{P}}$ to denote the prefix $\{0,1\}^k1\{*\}^{w-k-1}$.

Given a one-dimensional packet classifier f on $\{*\}^w$, we use $f_{\mathcal{P}}$ to denote a packet classifier on $\mathcal{P}$ such that for any $x \in \mathcal{P}$, $f_{\mathcal{P}}(x) = f(x)$, and we use $f_{\mathcal{P}}^d$ to denote a similar packet classifier on $\mathcal{P}$ with the additional restriction that the final decision is d.

We use $C(f_{\mathcal{P}})$ to denote the minimum cost of a packet classifier t that is equivalent to $f_{\mathcal{P}}$, and we use $C(f_{\mathcal{P}}^d)$ to denote the minimum cost of a packet classifier t' that is equivalent to $f_{\mathcal{P}}$ and the decision of the last rule in t' is d.

Given a one-dimensional packet classifier f on $\{*\}^w$ and a prefix $\mathcal{P}$ where $\mathcal{P} \subseteq \{*\}^w$, *$f$ is consistent on $\mathcal{P}$* if and only if $\forall x, y \in \mathcal{P}$, $f(x) = f(y)$.

Our dynamic programming solution to the weighted one-dimensional TCAM minimization problem is based on the following theorem. The proof of the theorem shows how to divide a problem into sub-problems and how to combine solutions to sub-problems into a solution to the original problem.

Theorem 4.1: Given a one-dimensional packet classifier f on $\{*\}^w$, a prefix $\mathcal{P}$ where $\mathcal{P} \subseteq \{*\}^w$, the set of all possible decisions $\{d_1, d_2, \cdots, d_z\}$ where each decision d_i has a cost w_{d_i} $(1 \leq i \leq z)$, we have that

$$C(f_{\mathcal{P}}) = \min_{i=1}^{z} C(f_{\mathcal{P}}^{d_i})$$

where each $C(f_{\mathcal{P}}^{d_i})$ is calculated as follows:
(1) If f is consistent on $\mathcal{P}$, then

$$C(f_{\mathcal{P}}^{d_i}) = \begin{cases} w_{f(x)} & \text{if } f(x) = d_i \\ w_{f(x)} + w_{d_i} & \text{if } f(x) \neq d_i \end{cases}$$

(2) If f is not consistent on $\mathcal{P}$, then

$$C(f_{\mathcal{P}}^{d_i}) = \min \begin{cases} C(f_{\underline{\mathcal{P}}}^{d_1}) + C(f_{\overline{\mathcal{P}}}^{d_1}) - w_{d_1} + w_{d_i}, \\ \ldots, \\ C(f_{\underline{\mathcal{P}}}^{d_{i-1}}) + C(f_{\overline{\mathcal{P}}}^{d_{i-1}}) - w_{d_{i-1}} + w_{d_i}, \\ C(f_{\underline{\mathcal{P}}}^{d_i}) + C(f_{\overline{\mathcal{P}}}^{d_i}) - w_{d_i}, \\ C(f_{\underline{\mathcal{P}}}^{d_{i+1}}) + C(f_{\overline{\mathcal{P}}}^{d_{i+1}}) - w_{d_{i+1}} + w_{d_i}, \\ \ldots, \\ C(f_{\underline{\mathcal{P}}}^{d_z}) + C(f_{\overline{\mathcal{P}}}^{d_z}) - w_{d_a} + w_{d_i} \end{cases}$$

□

Proof: (1) The base case of the recursion is when f is consistent on $\mathcal{P}$. In this case, the minimum cost prefix packet classifier in $\{f_{\mathcal{P}}\}$ is clearly $\langle \mathcal{P} \rightarrow f(x) \rangle$, and the cost of this packet classifier is $w_{f(x)}$. Furthermore, for $d_i \neq f(x)$, the minimum cost prefix packet classifier in $\{f_{\mathcal{P}}\}$ with decision d_i in the last rule is $\langle \mathcal{P} \rightarrow f(x), \mathcal{P} \rightarrow d_i \rangle$ where the second rule is redundant. The cost of this packet classifier is $w_{f(x)} + w_{d_i}$.

(2) If f is not consistent on $\mathcal{P}$, we divide $\mathcal{P}$ into $\underline{\mathcal{P}}$ and $\overline{\mathcal{P}}$. The crucial observation is that an optimal solution f^* to $\{f_{\mathcal{P}}\}$ is essentially an optimal solution f_1 to the sub-problem of minimizing $f_{\underline{\mathcal{P}}}$ appended with an optimal solution f_2 to the sub-problem of minimizing $f_{\overline{\mathcal{P}}}$. The only interaction that can occur between f_1 and f_2 is if their final rules have the same decision, in which case both final rules can be replaced with one final rule covering all of $\mathcal{P}$ with the same decision. Let d_x be the decision of the last rule in f_1 and d_y be the decision of the last rule in f_2. Then we can compose f^* whose last rule has decision d_i from f_1 and f_2 based on the following cases:
(A) $d_x = d_y = d_i$: In this case, f can be constructed by listing all the rules in f_1 except the last rule, followed by all the rules in f_2 except the last rule, and then the last rule $\mathcal{P} \rightarrow d_i$. Thus, $Cost(f) = Cost(f_1) + Cost(f_2) - w_{d_i}$.
(B) $d_x = d_y \neq d_i$: In this case, f can be constructed by listing all the rules in f_1 except the last rule, followed by all the rules in f_2 except the last rule, then rule $\mathcal{P} \rightarrow d_x$, and finally rule $\mathcal{P} \rightarrow d_i$ Thus, $Cost(f) = Cost(f_1) + Cost(f_2) - w_{d_x} + w_{d_i}$.
(C) $d_x \neq d_y, d_x = d_i, d_y \neq d_i$: We do not need to consider this case because $C(f_{\underline{\mathcal{P}}}^{d_i}) + C(f_{\overline{\mathcal{P}}}^{d_y}) = C(f_{\underline{\mathcal{P}}}^{d_i}) + (C(f_{\overline{\mathcal{P}}}^{d_y}) + w_{d_i}) - w_{d_i} \geq C(f_{\underline{\mathcal{P}}}^{d_i}) + C(f_{\overline{\mathcal{P}}}^{d_i}) - w_{d_i}$.
(D) $d_x \neq d_y, d_x \neq d_i, d_y = d_i$: Similarly, we do not need to consider this case.
(E) $d_x \neq d_y, d_x \neq d_i, d_y \neq d_i$: Similarly, we do not need to consider this case. ■

Figure 1 shows the illustration of a one-dimensional TCAM minimization problem, where the black bar denotes decision "accept" and the white bar denotes decision "discard". Figure 2 illustrates how the dynamic programming solution works on this example.

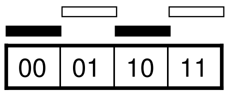

Fig. 1. An example one-dimensional TCAM minimization problem

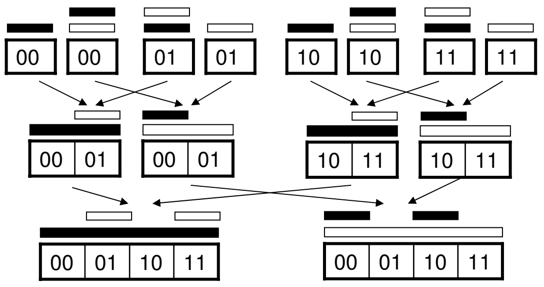

Fig. 2. Illustration of dynamic programming

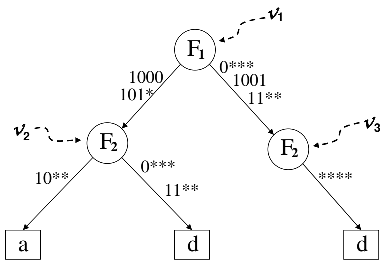

Fig. 3. A firewall decision diagram

V. Multi-dimensional TCAM Minimization

In this section, we present TCAM Razor, our algorithm for minimizing multi-dimensional prefix packet classifiers. A key idea behind TCAM Razor is processing one dimension at a time using the weighted one-dimensional TCAM minimization algorithm in Section IV to greedily identify a local minimum for the current dimension. Although TCAM Razor is not guaranteed to achieve a global minimum across all dimensions, it does significantly reduce the number of prefix rules in real-life packet classifiers. Due to space limitations, we omit the description of some optimizations that improve TCAM Razor's run-time performance.

A. Conversion to Firewall Decision Diagrams

To facilitate processing a packet classifier one dimension at a time, we first convert a given packet classifier to an equivalent *Firewall Decision Diagram* (FDD) [9].

A Firewall Decision Diagram (FDD) with a decision set DS and over fields $F_1, \cdots, F_d$ is an acyclic and directed graph that has the following five properties:

1) There is exactly one node that has no incoming edges. This node is called the *root*. The nodes that have no outgoing edges are called *terminal* nodes.
2) Each node v has a label, denoted $F(v)$, such that

$$F(v) \in \begin{cases} \{F_1, \cdots, F_d\} & \text{if } v \text{ is a nonterminal node,} \\ DS & \text{if } v \text{ is a terminal node.} \end{cases}$$

3) Each edge e:$u \rightarrow v$ is labeled with a nonempty set of integers, denoted $I(e)$, where $I(e)$ is a subset of the domain of u's label (*i.e.*, $I(e) \subseteq D(F(u))$).
4) A directed path from the root to a terminal node is called a *decision path*. No two nodes on a decision path have the same label.
5) The set of all outgoing edges of a node v, denoted $E(v)$, satisfies the following two conditions:
   a) *Consistency*: $I(e) \cap I(e') = \emptyset$ for any two distinct edges e and e' in $E(v)$.
   b) *Completeness*: $\bigcup_{e \in E(v)} I(e) = D(F(v))$. □

Figure 3 shows an example FDD over two fields F_1, F_2 where the domain of each field is $[0, 15]$. Note that in labelling terminal nodes, we use letter "a" as a shorthand for "$accept$" and letter "d" as a shorthand for "$discard$".

Given a packet classifier f_1, we can construct an equivalent FDD f_2 using the FDD construction algorithm in [15].

B. Multi-dimensional TCAM Minimization

We start the discussion of our greedy solution by examining the FDD in Figure 3. We first look at the subgraph rooted at node v_2. This subgraph can be seen as representing a one-dimension packet classifier over field F_2. We can use the weighted one-dimensional TCAM minimization algorithm in Section IV to minimize the number of prefix rules for this one-dimensional packet classifier. The algorithm takes the following 3 prefixes as input:

$$\begin{array}{ll} 10{*}{*} & \text{(with decision accept and cost 1),} \\ 0{*}{*}{*} & \text{(with decision discard and cost 1),} \\ 11{*}{*} & \text{(with decision discard and cost 1).} \end{array}$$

The one-dimensional TCAM minimization algorithm will produce a minimum (one-dimensional) packet classifier of two rules as shown in Table III.

Rule #	F_1	Decision
1	10**	accept
2	****	discard

TABLE III
A MINIMUM PACKET CLASSIFIER CORRESPONDING TO v_2 IN FIGURE 3

Similarly, from the subgraph rooted at node v_3, we can get a minimum packet classifier of one rule as shown in Table IV.

Rule #	F_1	Decision
1	****	discard

TABLE IV
A MINIMUM PACKET CLASSIFIER CORRESPONDING TO v_3 IN FIGURE 3

Next, we look at the root v_1. As shown in Figure 4, we view the subgraph rooted at v_2 as a decision with a multiplication factor or cost of 2, and the subgraph rooted at v_3 as another decision with a cost of 1. Thus, the graph rooted at v_1 can be thought of as a "virtual" one-dimensional packet classifier over field F_1 where each child has a multiplicative cost.

Now we are ready to use the one-dimensional TCAM minimization algorithm in Section IV to minimize the number

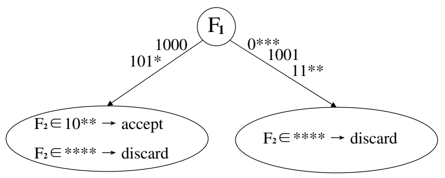

Fig. 4. “Virtual” one-dimensional packet classifier

of rules for this “virtual” one-dimensional packet classifier. The algorithm takes the following 5 prefixes and associated costs as input:

1000	(with decision v_2 and cost 2),
101$*$	(with decision v_2 and cost 2),
0 $***$	(with decision v_3 and cost 1),
1001	(with decision v_3 and cost 1),
11 $**$	(with decision v_3 and cost 1),

Running the weighted one-dimensional TCAM minimization algorithm on the above input will produce the “virtual” one-dimensional packet classifier of three rules as shown in Table V.

Rule #	F_1	Decision
1	1001	go to node v_3
2	10**	go to node v_2
3	****	go to node v_3

TABLE V

A MINIMUM PACKET CLASSIFIER CORRESPONDING TO v_1 IN FIGURE 3

Combining the “virtual” packet classifier in Table V and the two packet classifiers in Table III and IV, we get a packet classifier of 4 rules as shown in Table VI.

Rule #	F_1	F_2	Decision
1	1001	****	discard
2	10**	10**	accept
3	10**	****	discard
4	****	****	discard

TABLE VI

PACKET CLASSIFIER GENERATED FROM THE FDD IN FIGURE 3

C. Removing Redundant Rules

Next, we observe that rule r_3 in the packet classifier in Table VI is redundant. If we remove rule r_3, all the packets that used to be resolved by r_3 (that is, all the packets that match r_3 but do not match r_1 and r_2) are now resolved by rule r_4, and r_4 has the same decision as r_3. Therefore, removing rule r_3 does not change the semantics of the packet classifier. Redundant rules in a packet classifier can be removed using the algorithms in [16]. Finally, after removing redundant rules, we get a packet classifier of 3 rules from the FDD in Figure 3.

D. The Algorithm

To summarize, TCAM Razor, our multi-dimensional TCAM minimization algorithm, consists of the following four steps:

1) Convert the given packet classifier to an equivalent FDD.
2) Use the FDD reduction algorithm described in the next section to reduce the size of the FDD. This step will be explained in more detail in the next section.
3) Generate a packet classifier from the FDD in the following bottom up fashion. For every terminal node, assign a cost of 1. For a non-terminal node v with z outgoing edges $\{e_1, \cdots, e_z\}$, formulate a one-dimensional TCAM minimization problem as follows. For every prefix $\mathcal{P}$ in the label of edge e_j, $(1 \leq j \leq z)$, we set the decision of $\mathcal{P}$ to be j, and the cost of $\mathcal{P}$ to be the cost of the node that edge e_j points to. For node v, we use the weighted one-dimensional TCAM minimization algorithm in Section IV to compute a one-dimensional prefix packet classifier with the minimum cost. We then assign this minimum cost to the cost of node v. After the root node is processed, generate a packet classifier using the prefixes computed at each node in a depth first traversal of the FDD. The cost of the root indicates the total number of prefix rules in the resulting packet classifier.
4) Remove all the redundant rules from the resulting packet classifier.

VI. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness and efficiency of TCAM Razor on both real-life and synthetic packet classifiers. Note that in cases where TCAM Razor cannot produce smaller packet classifiers than redundancy removal alone, TCAM Razor will return the classifier produced by redundancy removal. Thus, TCAM Razor always performs at least as well as redundancy removal.

A. Methodology

We first define the metrics that we used to measure the effectiveness of TCAM Razor and the redundancy removal technique by Liu and Gouda [16]. In this paragraph, f denotes a packet classifier, S denotes a set of packet classifiers, and A denotes either TCAM Razor or the redundancy removal technique. We then let $|f|$ denote the number of rules in f, $A(f)$ denote the prefix classifier produced by applying A on f, and $Direct(f)$ denote the prefix classifier produced by applying direct range expansion on f. We define the following four metrics for assessing the performance of A on a set of classifiers S.

- The *average compression ratio* of A over $S = \frac{\Sigma_{f \in S} \frac{|A(f)|}{|Direct(f)|}}{|S|}$.
- The *total compression ratio* of A over $S = \frac{\Sigma_{f \in S}|A(f)|}{\Sigma_{f \in S}|Direct(f)|}$.
- The *average expansion ratio* of A over $S = \frac{\Sigma_{f \in S} \frac{|A(f)|}{|f|}}{|S|}$.
- The *total expansion ratio* of A over $S = \frac{\Sigma_{f \in S}|A(f)|}{\Sigma_{f \in S}|f|}$.

B. Effectiveness on Real-life Packet Classifiers

We first define a set RL of 17 real-life packet classifiers that we performed experiments on. We actually obtained 42 real-life packet classifiers from distinct network service providers that range in size from dozens to hundreds of rules. Although this collection of classifiers is diverse, some classifiers from the same network service provider have similar structure and exhibited similar results under TCAM Razor. To prevent this repetition from skewing the performance data, we divided the 42 packet classifiers into 17 structurally distinct groups, and we randomly chose one from each of the 17 groups to form the set RL.

1) Variable Ordering: The variable order that we used to convert a packet classifier into an equivalent FDD affects the effectiveness of TCAM Razor. There are $5! = 120$ different permutations of the five packet fields (source IP address, destination IP address, source port number, destination port number, and protocol type). We number these permutations from 1 to 120, and we use the notation TCAM Razor(i) to denote TCAM Razor using permutation i, and for a given packet classifier f, we use TCAM Razor(B) to denote TCAM Razor using the best of the 120 permutations for f.

A question that naturally arises is: *which variable order achieves the best average compression ratio?* To answer this question, for each permutation i, we computed the average compression ratio that TCAM Razor(i) achieved over RL. The results are shown in Figure 5. The maximum average compression ratio is 41.8%. Furthermore, more than half of the permutations have average compression ratios below 29.1%, and four permutations have average compression ratios below 18.3%. Of these four permutations, permutation 49 (*source IP address, protocol type, destination IP address, destination port number, source port number*) is the best with an average compression ratio of 18.2%.

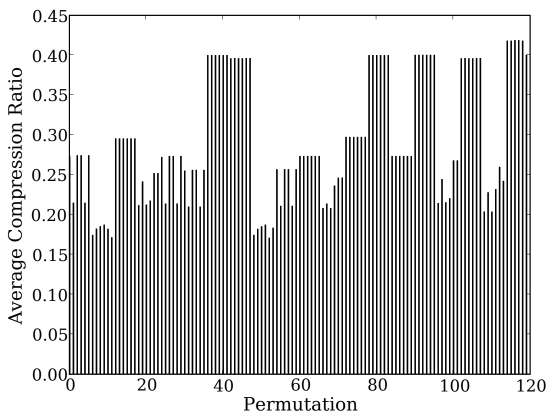

Fig. 5. The average compression ratio for each permutation

The next natural question to ask is: *is permutation 49 the best order for most packet classifiers?* The answer for RL is yes. In Figure 6, for each packet classifier in RL, we show the compression ratios of TCAM Razor(B), TCAM Razor(49), and redundancy removal. The results show that permutation 49 achieves almost the best compression ratio for each packet classifier group.

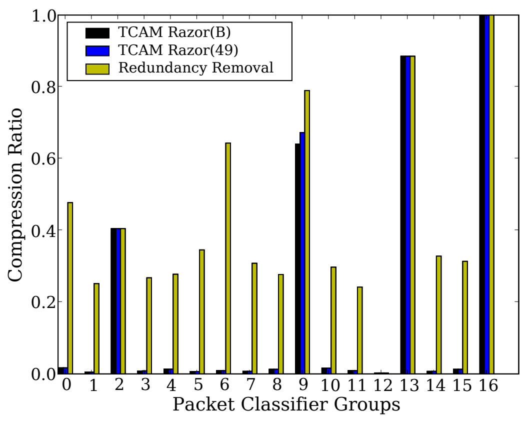

Fig. 6. Compression ratios of real-life packet classifier groups

2) Compression Ratio: Our experimental results clearly demonstrate that TCAM Razor outperforms just redundancy removal [16]. For example, the average compression ratios of TCAM Razor(49) and redundancy removal over RL are 18.2% and 41.8% respectively. Similarly, the total compression ratios of TCAM Razor(49) and redundancy removal over RL are 3.9% and 35% respectively. Figure 6 shows that TCAM Razor(49) significantly outperforms redundancy removal on 13 of the 17 real-life packet classifier groups. TCAM Razor(49) has a compression ratio of less than or equal to 1% on 8 of the 17 classifier groups in RL. Figure 7 shows the distribution of compression ratios achieved by TCAM Razor and redundancy removal alone on RL.

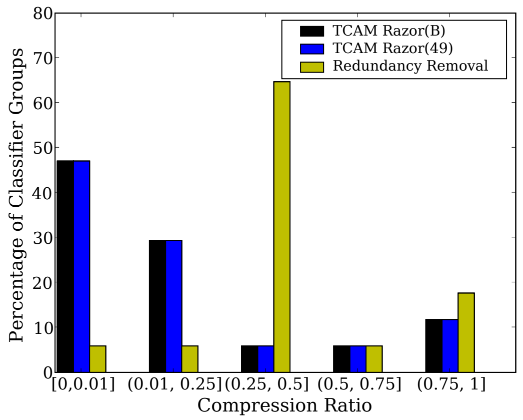

Fig. 7. Distribution of real-life packet classifiers by compression ratio

3) Expansion Ratio: We observe similar results for expansion ratio. The average expansion ratios for TCAM Razor(49), redundancy removal, and direction range expansion over RL are 0.754, 19.877, and 69.870, respectively. The total expansion ratio for TCAM Razor(49), redundancy removal, and direct range expansion over RL are 0.797, 7.147, and 20.414, respectively.

Figure 8 shows the distribution of expansion ratios for the following three algorithms: TCAM Razor(49), redundancy removal, and direct range expansion. Range expansion is a real issue as over 60% of our packet classifiers have an expansion ratio of over 50 if we use direct range expansion. The experimental data also suggests that TCAM Razor addresses the range expansion issue well as TCAM Razor(49) has an expansion ratio of at most 1 on 16 of the 17 real-life packet classifier groups in our experiments, and TCAM Razor(49) has an expansion ratio of 1.07 on the 17th real-life packet classifier.

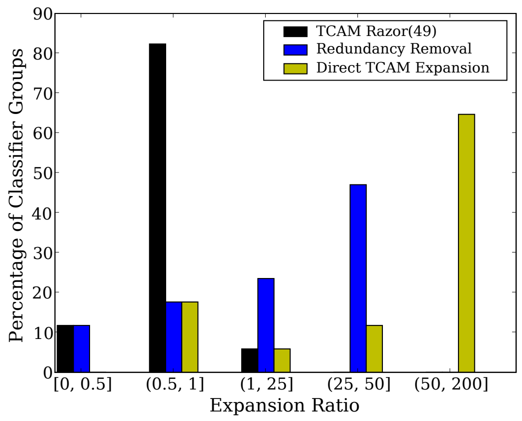

Fig. 8. Distribution of real-life packet classifiers by expansion ratio

C. *Comparison with Dong* et al. *[6]*

It is difficult to compare our results directly with those of Dong *et al.* [6] because we do not have access to their programs or the packet classifiers they experimented with. However, TCAM Razor(49) has a total compression ratio of 3.9% on our real-life packet classifiers. In contrast, Dong *et al.* reported a total compression ratio[1] of 54% on their real-life packet classifiers.

D. *Effectiveness on Synthetic Packet Classifiers*

Packet classifier rules are considered confidential due to security concerns. Thus, it is difficult to get many real-life packet classifiers to experiment with. To address this issue and further evaluate the performance of TCAM Razor, we generated SYN, a set of synthetic packet classifiers of 18 sizes, where each size has 100 independently generated classifiers.

Every predicate of a rule in our synthetic packet classifiers has five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. We first randomly generated a list of values for each field. For IP addresses, we generated a random class C address; for ports we generated a random range; for protocols, we generated a random protocol number. Given these lists, we generated a list of predicates by taking the cross product of all these lists. We added a final default predicate to our list. Finally, we randomly assigned one of two decisions, accept or deny, to each predicate to make a complete rule.

Distributions of compression ratios and expansion ratios over SYN are shown in Figures 9 and 10. The average compression ratio of TCAM Razor(49) over SYN is 4.6%, the average expansion ratio of TCAM Razor(49) over SYN is 8.737, the total compression ratio of TCAM Razor(49) over SYN is 1.6%, and the total expansion ratio of TCAM Razor(49) over SYN is 3.082.

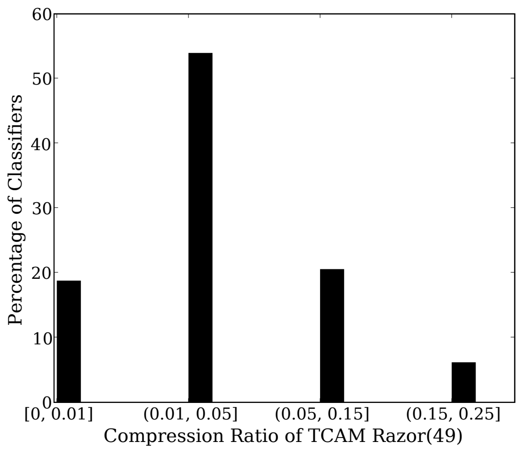

Fig. 9. Distribution of synthetic packet classifiers by compression ratio

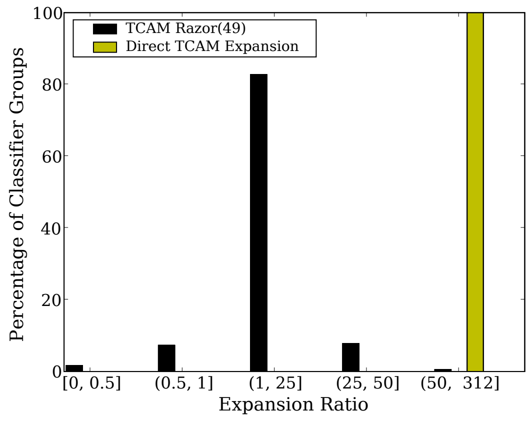

Fig. 10. Distribution of synthetic packet classifiers by expansion ratio

E. *Efficiency of TCAM Razor*

We implemented TCAM Razor using Visual Basic on Microsoft .Net framework 2.0. In our experiments, we first ran TCAM Razor on real-life packet classifiers, and then we stress tested TCAM Razor on a large number of big synthetic packet

[1]By clarifying with the authors of [6], the term "average compression ratio" in [6] is actually what we define as "total compression ratio" in this paper.

classifiers. Our experiments were carried out on a desktop PC running Windows XP with 1G memory and a single 2.2 GHz AMD Opteron 148 processor. Table VII shows the total running time of TCAM Razor(49) for three representative packet classifiers. Figure 11 displays the average total running time of TCAM Razor(49) on our synthetic packet classifiers as a function of the number of original rules along with the standard deviation.

Number of Original Rules	TCAM Razor Running Time (seconds)
42	0.2
87	0.9
661	31.9

TABLE VII
SAMPLE RUNNING TIME DATA FOR REAL-LIFE PACKET CLASSIFIERS

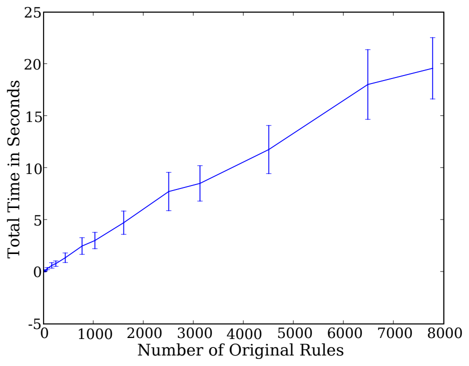

Fig. 11. Total runtime vs. number of original rules

VII. Conclusions

TCAMs have become the de facto industry standard for packet classification. However, as the rules in packet classifiers grow in number and complexity, the viability of TCAM-based solutions is threatened by the problem of range expansion. In this paper, we propose TCAM Razor, a systematic approach to minimizing TCAM rules for packet classifiers. While TCAM Razor does not always produce optimal packet classifiers, in our experiments with 17 structurally distinct real-life packet classifier groups, TCAM Razor reduced the number of TCAM entries needed by an average of 81.8% percent and a total of 96.1%. In fact, TCAM Razor experienced no expansion for 16 of the 17 real-life packet classifier groups. While it is difficult to perform a direct comparison with Deng *et al.*'s approach [6], it appears that TCAM Razor performs significantly better with a total compression ratio of 3.9% as compared with a total compression ratio of 54%. Finally, unlike other solutions that require modifying TCAM circuits or packet processing hardware, TCAM Razor can be deployed today by network administrators and ISPs to cope with range expansion.

References

[1] Cypress semiconductor corp. content addressable memory. http://www.cypress.com/.

[2] A guide to search engines and networking memory. *http://www.linleygroup.com/pdf/NMv4.pdf*, November 2006.

[3] D. A. Applegate, G. Calinescu, D. S. Johnson, H. Karloff, K. Ligett, and J. Wang. Compressing rectilinear pictures and minimizing access control lists. In *Proceedings of the Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2007.

[4] F. Baboescu, S. Singh, and G. Varghese. Packet classification for core routers: Is there an alternative to CAMs? In *Proceedings of IEEE INFOCOM*, 2003.

[5] F. Baboescu and G. Varghese. Scalable packet classification. In *Proceedings of ACM SIGCOMM*, pages 199–210, 2001.

[6] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla. Packet classifiers in ternary CAMs can be smaller. In *Proceedings of SIGMETRICS*, pages 311–322, 2006.

[7] R. Draves, C. King, S. Venkatachary, and B. Zill. Constructing optimal IP routing tables. In *Proceedings of IEEE INFOCOM*, pages 88–97, 1999.

[8] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *Proceedings of 19th IEEE INFOCOM*, Mar. 2000.

[9] M. G. Gouda and A. X. Liu. Structured firewall design. *Computer Networks Journal*, 51(4):1106–1120, March 2007.

[10] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proceedings of ACM SIGCOMM*, pages 147–160, 1999.

[11] P. Gupta and N. McKeown. Packet classification using hierarchical intelligent cuttings. In *Proceedings of Hot Interconnects VII*, Aug. 1999.

[12] P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, 15(2):24–32, 2001.

[13] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proceedings of ACM SIGCOMM*, pages 203–214, 1998.

[14] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with ternary cams. In *Proceedings of the ACM SIGCOMM*, pages 193 – 204, August 2005.

[15] A. X. Liu and M. G. Gouda. Diverse firewall design. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN-04)*, pages 595–604, June 2004.

[16] A. X. Liu and M. G. Gouda. Complete redundancy detection in firewalls. In *Proceedings of 19th Annual IFIP Conference on Data and Applications Security, LNCS 3654, S. Jajodia and D. Wijesekera Ed., Springer-Verlag*, pages 196–209, August 2005.

[17] H. Liu. Efficient mapping of range classifier into ternary-cam. In *Proceedings of the Hot Interconnects*, pages 95– 100, 2002.

[18] M. H. Overmars and A. F. van der Stappen. Range searching and point location among fat objects. *Journal of Algorithms*, 21(3):629–656.

[19] L. Qiu, G. Varghese, and S. Suri. Fast firewall implementations for software-based and hardware-based routers. In *Proceedings the 9th International Conference on Network Protocols (ICNP)*, 2001.

[20] S. Singh, F. Baboescu, G. Varghese, and J. Wang. Packet classification using multidimensional cutting. In *Proceedings of ACM SIGCOMM*, 2003.

[21] E. Spitznagel, D. Taylor, and J. Turner. Packet classification using extended tcams. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, pages 120– 131.

[22] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *Proceedings of ACM SIGCOMM*, pages 191–202, 1998.

[23] S. Suri, T. Sandholm, and P. Warkhede. Compressing two-dimensional routing tables. *Algorithmica*, 35:287–300, 2003.

[24] D. E. Taylor. Survey & taxonomy of packet classification techniques. *ACM Computing Surveys*, 37(3):238–275, 2005.

[25] J. van Lunteren and T. Engbersen. Fast and scalable packet classification. *IEEE Journals on Selected Areas in Communications*, 21(4):560– 571, 2003.

[26] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high speed IP routing lookups. In *Proceedings of ACM SIGCOMM*, pages 25–36, September 1997.

[27] T. Y. C. Woo. A modular approach to packet classification: Algorithms and results. In *Proceedings of IEEE INFOCOM*, pages 1213–1222, 2000.

[28] F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz. Ssa: A power and memory efficient scheme to multi-match packet classification. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 105–113, October 2005.