

# Bidirectional Range Extension for TCAM-Based Packet Classification

Yan Sun and Min Sik Kim

School of Electrical Engineering and Computer Science,  
Washington State University,  
Pullman, Washington 99164-2752, U.S.A.  
{ysun,msk}@eecs.wsu.edu

**Abstract.** Packet classification is a fundamental task for network devices such as edge routers, firewalls, and intrusion detection systems. Currently, most vendors use Ternary Content Addressable Memories (TCAMs) to achieve high-performance packet classification. TCAMs use parallel hardware to check all rules simultaneously. Despite their high speed, TCAMs have a problem in dealing with ranges efficiently. Many packet classification rules contain range specifications, each of which needs to be translated into multiple prefixes to store in a TCAM. Such translation may result in an exponential increase in the number of required TCAM entries. In this paper, we propose a bidirectional range extension algorithm to solve this problem. The proposed algorithm uses at most two TCAM entries to represent a range, and can be pipelined to deal with multiple range fields in a packet header. Since this algorithm assumes a non-redundant rule set, i.e., no range overlap between different rules, which can be obtained by applying our previous work on redundancy removal in TCAM using a tree representation of rules. Our experiments show a more than 75% reduction in the number of TCAM entries by applying the bidirectional range extension algorithm to real-world rule sets.

**Keywords:** packet classification, TCAM, bidirectional range extension.

## 1 Introduction

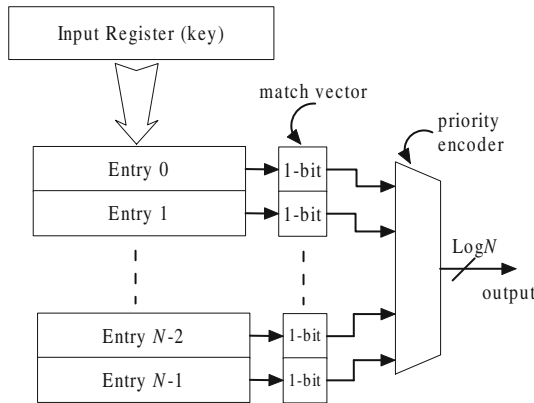
There are a number of network services that require packet classification, such as policy-based routing, firewalls, provision of differentiated qualities of service, and traffic billing. In each case, it is necessary to determine which flow an arriving packet belongs to, for example, where to forward it, whether to forward or filter it, what class of service it should receive, or how much should be charged for transporting it. As packet classification has been widely deployed on the Internet, demand for efficient packet classification grows. The function of a packet classification system is to map each packet to a decision according to a sequence of rules, which is called a packet classifier. The rules specified in a packet classifier may or may not be mutually exclusive; two rules may overlap. When it happens with no explicit priorities specified, we follow the convention that a

**Table 1.** A Simple Header Rule Set

Rule	Type	Source IP	Destination IP	Source Port	Destination Port	Decision
$r_1$	TCP	*	192.168.0.0/16	<1024	*	accept
$r_2$	TCP	*	192.168.14.1	*	139	discard
$r_3$	UDP	192.168.0.0/16	*	*	700:900	accept
$r_4$	TCP	*	*	*	*	discard

rule closer to the top of the list takes priority. Table 1 shows a simple packet classifier of four rules.

Perhaps the most popular method for high-speed packet classification in practice is to use a Ternary Content Addressable Memory (TCAM) [1]. A TCAM is a memory chip where each entry can store a packet classification rule in ternary form. It stores data patterns in the form of (value, bit mask) pairs. A query key is simultaneously compared against all the patterns stored in a TCAM. A key  $q$  is said to match a stored pattern  $(v, m)$  if  $q \& m = v \& m$ , where “&” is the bit-wise logical AND operator. 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 through a priority encoder. Thus, it takes  $O(1)$  time to find the decision for any given packet. Because of their high speed, TCAMs have become the industrial standard for high speed packet classification [2]. The architecture of a TCAM used in packet classification is shown in Fig. 1.



**Fig. 1.** TCAMs used in the packet classifications

A key (Protocol, Source IP address, Destination IP address, Source Port, and Destination Port) is stored in the input register and the rules are stored in the TCAM entries. The key compares with all the entries in parallel and the results are stored in the match vector, where 1’s represent that the corresponding entries

match the key, and the priority encoder chooses the match with the highest priority. At last, the output signal is used to find the corresponding action.

Despite their high speed, TCAMs have two major drawbacks when used in packet classifiers. First, they consume a large amount of power and have high hardware cost. Thus, their capacity in packet classifiers is often limited. Second, they are inefficient when applied to packet classifiers with port number ranges, because TCAMs can only store rules in ternary form, which means that port numbers need to be converted to one or more prefixes before being stored in TCAMs. This may lead to a significant increase in the number of TCAM entries needed to encode a rule. For example, 30 prefixes are needed to represent a single source port range [1, 65534], and 20 prefixes are needed to represent a destination port range [1, 2046]. Thus,  $30 \times 20 = 600$  TCAM entries are required to represent a single rule with these two ranges. We observe that packet classifiers typically have at most one port range in each rule, and rules specifying two port ranges are rare. However, a small number of such rules can consume most of the TCAM entries and the number of such rules is increasing. Therefore, minimizing the number of required TCAM entries is crucial.

In this paper, we first introduce our previous work on redundancy removal in TCAM using a tree representation of rules [3], which removes all redundancy in the original rule set to make sure each packet matches and only matches a single new rule. Then we propose a bidirectional range extension algorithm to solve the range explosion problem based on the non-redundant rule set. The proposed algorithm uses at most two TCAM entries to represent a range, and can be pipelined to deal with multiple range fields in a packet header. In our experiments, we achieve a total reduction of 84.27% in the number of TCAM entries consumption.

The remainder of the paper is organized as follows. Section 2 presents previous work related to this paper. Section 3 discusses our previous work on removing redundancies in packet classification rules, and Section 4 proposes our bidirectional range extension algorithm to reduce the number of TCAM entries. Then the experimental results are presented in Section 5. Finally, we conclude in Section 6.

## 2 Related Work

Previous work exploring solutions to deal with the range expansion problem falls into two major categories: hardware-based solutions, which require changing TCAM hardware circuits, and software-based solutions, which do not require such changes. Below, we review previous work in these two categories.

*Hardware-based solutions:* The basic idea of hardware-based solutions is to modify TCAM circuits and architecture [2,4,5,6,7]. For example, van Lunteren et al. proposed a method of adding comparators at each entry to better accommodate range matching in packet classifiers [6]. While this allows to use TCAMs more efficiently, any solution from this research line has some drawbacks such as the cost of hardware modification.

*Software-based solutions:* Software-based solutions are more likely to be adopted by networking vendors and ISPs because they do not require changing TCAM hardware or existing packet classification systems. Many software-based solutions have been proposed [8,9,10,11,12,13,14,15,3] to reduce the TCAM entry consumption. Their basic idea is to preprocess ranges that appear in a packet classifier or convert a given packet classifier to another semantically-equivalent packet classifier that requires fewer TCAM entries, and then store the new rule set in a TCAM. Hence, the TCAM circuits need not be modified to implement range storage. Although these methods can typically achieve a 40–60% reduction in the number of TCAM entries by reducing redundancy in the rule set or combining multiple TCAM entries into a single TCAM entry, more reduction is desirable because of the low capacity and high cost of TCAMs. Our work first reduces the redundancy in the rule set, and then extends the range in both directions, upward and downward, storing a single range into at most two TCAM entries. To the best of our knowledge, this article is the first attempt at using a range extension method to reduce the TCAM entries consumption.

### 3 Redundancy Removal Tree

Packet classifiers usually check the following five fields in each packet header: protocol type, source port number, destination port number, source IP address, and destination IP address. When a TCAM is used to implement a packet classifier, all rules stored in TCAM entries must be represented as exact binary values or binary values with wildcard bits. However, in a typical packet classification rule, some fields such as source and destination port numbers are represented as integer ranges rather than exact values or binary prefix values. Thus, we need to convert a rule with fields represented as integer ranges into one or more binary prefixes, which may lead to range expansion. During the process of range expansion, each field of a rule should be expanded separately. For example, if a 3-bit field of a rule is  $[1, 6]$ , the corresponding minimum set of prefixes covering the range includes 001, 01\*, 10\*, and 110. In the worst-case, range expansion of a  $w$ -bit integer range yields  $2^w - 2$  prefixes [16]. The next step to the range expansion is to compute the cross-product of obtained prefix sets, resulting in an exponential increase of the number of prefixes needed to replace a single rule.

The problem can be mitigated by removing redundancy in the original rule set. Two rules in a packet classifier may overlap, which means that one packet may match two or more rules. Besides, two rules in a packet classifier may conflict with each other. In other words, two overlapping rules may have different decisions. Many packet classifiers resolve conflicts by choosing the first match, which has a higher priority. In firewalls, typical decisions include “accept,” “discard,” “accept with logging,” and “discard with logging.”

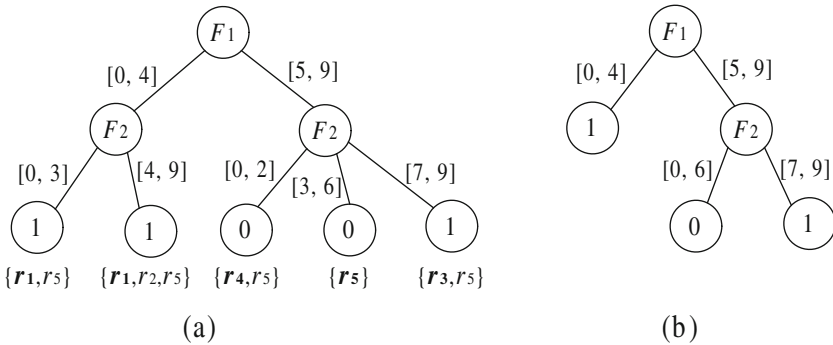
Our goal is to reduce redundancy in a given packet classifier by removing redundant rules and overlapping parts. To achieve this goal, we build a Minimum Range Tree  $T$  for a packet classifier  $C$ :  $(r_1, r_2, \dots, r_n)$  over fields  $F_1, \dots, F_d$ . The tree  $T$  must satisfy the followings:

- The height of the tree is equal to the number of fields in the packet classifier.
- Edges of each depth of the tree store the ranges of the corresponding field. All edges in the same depth cover the whole range of the field, and there is no overlap between any pair of them.
- A directed path from a leaf node to the root is called a *decision path*. For a given packet, the tree has exactly one matched decision path.
- Each leaf node is labeled with the decision associated with the corresponding decision path.

Fig. 3(a) shows a range tree for the simple packet classifier in Fig. 2. In this example, we assume every packet has only two fields,  $F_1$  and  $F_2$ , and the domain of each field is  $[0, 9]$ .

$$\begin{aligned}
 r_1 &: F_1 \in [0, 4] \wedge F_2 \in [0, 9] \rightarrow \text{accept} \\
 r_2 &: F_1 \in [0, 4] \wedge F_2 \in [4, 9] \rightarrow \text{accept} \\
 r_3 &: F_1 \in [5, 9] \wedge F_2 \in [7, 9] \rightarrow \text{accept} \\
 r_4 &: F_1 \in [5, 9] \wedge F_2 \in [0, 2] \rightarrow \text{discard} \\
 r_5 &: F_1 \in [0, 9] \wedge F_2 \in [0, 9] \rightarrow \text{discard}
 \end{aligned}$$

**Fig. 2.** A simple packet classifier



**Fig. 3.** Constructing a Minimum-Range-Tree for the Packet Classifier in Fig. 1

In Fig. 3, each edge represents a range in the corresponding field, and we use number 1 as a shorthand for “accept” and number 0 as a shorthand for “discard” in labeling leaf nodes. We first build a tree as in Fig. 3(a) according to the packet classifier in Fig. 2, then we combine two neighboring leaf nodes if they have the same decision and share the same parent node. The Minimum-Range-Tree is shown in Fig. 3(b). Now we get three new non-redundant rules based on the Minimum-Range-Tree. Please see previous work [3] for details.

## 4 Proposed Algorithm

After removing redundancies with the minimum range tree, there remains no overlap between different rules, and thus we can sort all the new rules by their ranges in a field. However, this does not solve the range explosion problem; it only mitigates it. Still, a single range may need multiple TCAM entries to represent itself. In the worst case,  $2m - 2$  entries are needed to store a single  $m$ -bit range. For a 16-bit port range, it may need 30 TCAM entries. In order to further reduce the number of TCAM entries, we propose a bidirectional range extension, which only needs at most two TCAM entries to represent a single range regardless of the length of the range.

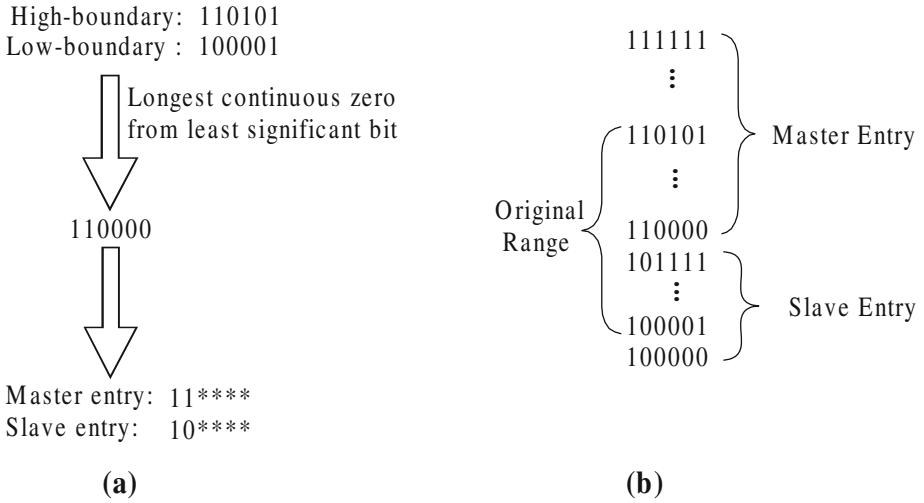
### 4.1 Bidirectional Range Extension

Given a list of ranges, we assume that every value belongs to exactly one range, that a range appearing earlier in the list has a higher priority than a range appearing later, and that the output of range matching for each range is binary, either 1 or 0. Note that applying the minimum range tree algorithm described in Sec. 3 and sorting resulting ranges by the boundary values yield a list of ranges satisfying these assumptions. Assuming these, we take the following steps to reduce the number of required TCAM entries.

We divide each range into two parts, each of which is represented as a single prefix. One part is called an upward extension and the other a downward extension. It is demonstrated in Fig. 4. Given a range, e.g., [100001, 110101], find a number  $N$  that belongs to the range and has the most consecutive 0's starting at the least significant bit (LSB). In our example,  $N$  will be 110000. Convert all the rightmost consecutive 0's of  $N$  into "don't care" bits. The result will be the upward extension (master entry). If  $N$  is equal to the lower bound of the original range, this range does not need a downward extension. Otherwise, the numeric prefix (excluding "don't care" bits) of the upward extension minus 1 (10 in our example), followed by the same number of "don't care" bits will be the downward extension (slave entry) as shown in Fig. 4(a). We use the set of all upward extensions as the master set and the set of downward extensions as the slave set. Note that the master set has the same number of entries as the number of ranges in the original list. On the other hand, the slave set only contains entries for those ranges where the original range is not a subset of the upward extension.

Below, We prove that the original range is a subset of the master and slave entries combined.

Proof: Assume that the range is  $[N_L, N_H]$ . Suppose  $N_L = \sum_{i=0}^{n-1} b_i^L 2^i$  and  $N_H = \sum_{i=0}^{n-1} b_i^H 2^i$ , where  $b_i^L, b_i^H \in \{0, 1\}$ . Let  $N$  be the number in  $[N_L, N_H]$  that has the longest consecutive zeros ( $n_0$  bits) starting at LSB. Thus, we have  $N = 2^{n_0} + \sum_{i=n_0+1}^{n-1} b_i 2^i$ , where  $b_i \in \{0, 1\}$ . Then the master entry is a range  $[N, N_{\text{master}}]$ , where  $N_{\text{master}} = N + \sum_{i=0}^{n_0-1} 2^i$  with at least consecutive  $n_0 + 1$  1's starting at LSB. Therefore,  $N_{\text{master}} + 1 = 2^{n_0+1} + \sum_{i=n_0+1}^{n-1} b_i 2^i$ , which has at



**Fig. 4.** An example of how to divide a range and generate master/slave entries

least  $n_0 + 1$  consecutive 0’s starting at LSB, or more 0’s than  $N$ , and thus should not be in  $[N_L, N_H]$ . Therefore,

$$N_H \leq N_{\text{master}} . \tag{1}$$

Similarly, the slave entry, if we have one, is a range  $[N_{\text{slave}}, N - 1]$ , where  $N_{\text{slave}} = N - 2^{n_0} = \sum_{i=n_0+1}^{n-1} b_i 2^i$ . Note that  $N_{\text{slave}}$  has at least  $n_0 + 1$  consecutive 0’s starting at LSB, or more 0’s than  $N$ , and thus should not be in  $[N_L, N_H]$ . Therefore,

$$N_L > N_{\text{slave}} . \tag{2}$$

By Eq. 1 and Eq. 2,  $[N_L, N_H]$  is a subset of  $[N_{\text{slave}}, N_{\text{master}}]$ .

Note that each of the master and slave entries can be represented by a single prefix, and thus by a single TCAM entry.

### 4.2 Matching Using Extended Ranges

A field of an incoming packet is compared against both the master and slave sets; each set is stored in a separate TCAM. After knowing the decision from the master set, we need to find out whether the matched entry has a corresponding entry in the slave set. If there is no such entry, the decision in the master set becomes the final decision; otherwise, we need to consider the corresponding slave entry. In the slave set, we obtain the decision and the low-boundary ( $N_L$ ) of the corresponding entry. If both decisions, one from the master set and the other from the slave set, are identical, that becomes the final decision; otherwise we compare the low-boundary from the slave set with the input key. If the low-boundary is smaller than the key, we choose the decision from the slave set; otherwise we choose the decision from the master set.

For the worst case, the single interval [1, 65534] requires only 2 TCAM entries in our algorithm instead of 30 entries, which reduces about 93.3%.

## 5 Simulation Results

### 5.1 Experimental Results on Minimum Range Tree

For experiments, we collected rules from actual packet classifiers. Because rule sets vary across different applications, we gathered as many rules as we could and then randomly selected one thousand rules from them. For the Minimum Range Tree, we compared the number of TCAM entries used by the original rules and the number of TCAM entries used after removing redundancies using the Minimum Range Tree. The simulation result showed that our algorithm reduces 66.4% of TCAM entries in total [3].

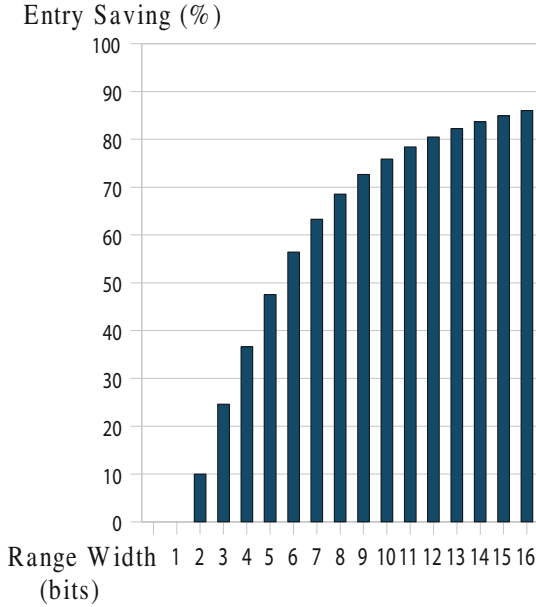
### 5.2 Analysis of Bidirectional Range Extension

In this subsection we analyze the performance of the proposed bidirectional range extension algorithm on a single range, such as the destination port range. Given the number of bits for the field, we generate every possible range whose length is greater than 1 with the same probability. Thus, the generated rule set may include ranges  $[0, 1]$ ,  $[0, 2]$ ,  $\dots$ ,  $[0, 2^n - 1]$ ,  $[1, 2]$ ,  $[1, 3]$ ,  $\dots$ ,  $[2^n - 2, 2^n - 1]$ , where  $n$  is the number of bits of this field. And the simulation results are shown in Table 2.

**Table 2.** Analysis of a Single Range

Range length (bits)	# of TCAM entries with range extension	# of TCAM entries w/o range extension	Saving (%)
1	1	1	0.00
2	9	10	10.00
3	49	65	24.62
4	225	355	36.62
5	961	1831	47.52
6	3969	9103	56.40
7	16129	43935	63.29
8	65025	206911	68.57
9	261121	955007	72.66
10	1046529	4335871	75.86
11	4190209	19421695	78.43
12	16769025	86033407	80.51
13	67092481	377595903	82.23
14	268402689	1644400639	83.68
15	1073676289	7114039295	84.91
16	4294836225	30602706943	86.00

The percentages of entries saved with different field widths are shown in Fig. 5. We can see that the percentage of entries saved by our algorithm increases as the field width increases, and we can save 86.00% of TCAM entries of the 16-bit port number field. For the real-world rule sets we collected<sup>1</sup>, our algorithm can save 77.47% and 74.81% entries for the destination port field and source port field, respectively, excluding non-range rules. For a single 16-bit port number, our approach reduces the average number of entries from 14.25 to 2.



**Fig. 5.** The Entries saved with Different Field Width

For a rule with both destination port and source port ranges, our algorithm can further save TCAM entries because of the multiplication effect. For the scenarios shown in Table 2, our algorithm saves 98.04% of TCAM entries for randomly generated rules with both destination port and source port fields, and saves 82.78% for the ranges included in the real-world rule sets. The difference between them is mainly caused by the fact that some popular ranges, such as [0, 1023], can be represented by a single prefix in real-world rule sets. If we include rules without ranges in the simulation, our algorithm can save 53.18% TCAM entries for the real-world rule sets.

From both redundancy removal using the Minimum Range Tree and the bidirectional range extension, we can reduce 84.27% of entries for the real-world rule sets with additional circuits such as two adders and a little latency.

<sup>1</sup> <http://www.routeviews.org/>

## 6 Conclusion

In this paper, we proposed a bidirectional range extension algorithm to solve the range explosion problem in TCAM. The proposed algorithm assumes a non-redundant rule set, which can be achieved by previous work. Our algorithm first divides a range into two ranges, and then extends the ranges upward and downward to make each extended range consumes at most one TCAM entry. The result is that each range consumes at most two TCAM entries. Our algorithm significantly reduces the number of TCAM entries needed by a packet classifier. In our experiments, after removing redundancies, we observed a reduction of 86.00% on average in the number of TCAM entries, and an overall reduction of 84.27% for real-world rule.

## References

1. Pagiartzis, K., Sheikholeslami, A.: Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits* 41(3), 712–727 (2006)
2. Karthik, L., Anand, R., Srinivasan, V.: Algorithms for advanced packet classification with ternary CAMs. In: *Proceedings of ACM SIGCOMM '05, August 2005*, pp. 193–204 (2005)
3. Sun, Y., Kim, M.S.: Tree-based minimization of TCAM entries for packet classification. In: *Proceedings of the 7th IEEE Consumer Communications and Networking Conference (January 2010)*
4. Huan, L.: Efficient mapping of range classifier into ternary-CAM. In: *Proceedings of the 10th Symposium on High Performance Interconnects, August 2002*, pp. 95–100 (2002)
5. Lunteren, J., Engbersen, T.: Fast and scalable packet classification. *IEEE Journal on Selected Areas in Communications* 21(4), 560–571 (2003)
6. Ed, S., David, T., Jonathan, T.: Packet classification using extended TCAMs. In: *Proceedings of the 11th IEEE International Conference on Network Protocols, November 2003*, pp. 120–131 (2003)
7. Faezipour, M., Nourani, M.: CAM01-1: A customized TCAM architecture for multi-match packet classification. In: *IEEE Global Telecommunications Conference, November 2006*, pp. 1–5 (2006)
8. Sumeet, S., Florin, B., George, V., Jia, W.: Packet classification using multidimensional cutting. In: *Proceedings of ACM SIGCOMM '03, August 2003*, pp. 213–224 (2003)
9. Liu, A.X., Meiners, C.R., Zhou, Y.: All-match based complete redundancy removal for packet classifiers in TCAMs. In: *Proceedings of the 27th IEEE INFOCOM, April 2008*, pp. 111–115 (2008)
10. Applegate, D.A., Calinescu, G., Johnson, D.S., Karloff, H., Ligett, K., Wang, J.: Compressing rectilinear pictures and minimizing access control lists. In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, January 2007*, pp. 1066–1075 (2007)
11. Qunfeng, D., Suman, B., Jia, W., Dheeraj, A., Ashutosh, S.: Packet classifiers in ternary CAMs can be smaller. In: *Proceedings of ACM SIGMETRICS/Performance 2006, June 2006*, pp. 311–322 (2006)

12. Meiners, C.R., Liu, A.X., Torng, E.: Topological transformation approaches to optimizing TCAM-based packet classification systems. In: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems, pp. 73–84 (2009)
13. Bremler-Barr, A., Hendler, D.: Space-efficient TCAM-based classification using gray coding. In: Proceedings of the 26th IEEE International Conference on Computer Communications, May 2007, pp. 6–12 (2007)
14. Che, H., Wang, Z., Zheng, K., Liu, B.: DRES: Dynamic range encoding scheme for TCAM coprocessors. *IEEE Transactions on Computers* 57(7), 902–915 (2008)
15. Pao, D., Li, Y.K., Zhou, P.: Efficient packet classification using tcams. *Computer Networks: The International Journal of Computer and Telecommunications Networking* 50(18), 3523–3535 (2006)
16. Gupta, P., Mckeown, N.: Algorithms for packet classification. *IEEE Network* 15(2), 24–32 (2001)