DPPC-RE: TCAM-Based Distributed Parallel Packet Classification with Range Encoding

Kai Zheng, *Student Member*, *IEEE*, Hao Che, *Member*, *IEEE*,
Zhijun Wang, Bin Liu, *Member*, *IEEE*, and Xin Zhang

Abstract—Packet classification has been a critical data path function for many emerging networking applications. An interesting approach is the use of Ternary Content Addressable Memory (TCAM) to achieve deterministic, high-speed packet classification performance. However, apart from high cost and power consumption, due to slow growing clock rate for memory technology, in general, the traditional single TCAM-based solution has difficulty to keep up with fast growing line rates. Moreover, the TCAM storage efficiency is largely affected by the need to support rules with ranges or range matching. In this paper, a distributed TCAM scheme that exploits chip-level-parallelism is proposed to greatly improve the throughput performance. This scheme seamlessly integrates with a range encoding scheme which not only solves the range matching problem, but also ensures a balanced high throughput performance. A thorough theoretical worst-case analysis of throughput, processing delay, and power consumption, as well as the experimental results show that the proposed solution can achieve scalable throughput performance matching up to OC768 line rate or higher. The added TCAM storage overhead is found to be reasonably small for the five real-world classifiers studied.

Index Terms—Packet classification, range matching, TCAM.

1 Introduction

PACKET Classification has wide applications in networking devices to support firewall, access control list (ACL), and quality of service (QoS) in access, edge, and/or core networks. It may involve various matching conditions, e.g., longest prefix matching (LPM), exact matching, and range matching, making it a complicated pattern matching issue. Moreover, since it lies in the critical data path (i.e., it has to act upon each and every packet at wire speed), packet classification may create potential bottlenecks in the router data path, particularly for high-speed interfaces. For example, at OC192 (10Gbps) full line rate, a line card needs to process about 31.25 *Million Packets Per Second*[1] (Mpps) in the worst case when minimum sized packets (40 bytes each) arrive back-to-back. As the aggregate line rate to be supported by a line card is moving toward OC768 (i.e., approximately 125[2] Mpps in the worst case) [23], it poses significant challenges for the development of packet classification mechanisms that match wire speed forwarding performance.

1. $10Gbps/(40^*8b) \approx 31.25Mpps$.
2. $40Gbps/(40^*8b) \approx 125Mpps$.

• K. Zheng and B. Liu are with the Department of Computer Science, Tsinghua University, Beijing, PR China 100084.
E-mail: zk01@mails.tsinghua.edu.cn, liub@tsinghua.edu.cn.
• H. Che is with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019.
E-mail: hche@cse.uta.edu.
• Z. Wang is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. E-mail: cszjwang@comp.polyu.edu.hk.
• X. Zhang is with the Department of Automation, Tsinghua University, Beijing, PR China 100084. E-mail: z-x02@mails.tsinghua.edu.cn.

Manuscript received 1 June 2005; revised 16 Jan. 2006; accepted 16 Feb. 2006; published online 21 June 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0179-0605.

The existing algorithmic approaches, including geometric algorithms based on hierarchical tries [1], [2], [3], [4], [5] and most heuristic algorithms [6], [7], [8], [9], generally require a nondeterministic number of memory accesses per lookup, making it difficult to use pipeline to hide the memory access latency and, hence, limiting the throughput performance. The algorithmic approach in [25] allows deterministic time for lookup, i.e., $O(\log W)$ time per lookup, where W is the length of the search key. However, it is not fast enough to match high-speed interface and the storage requirement depends on the parameter chosen. In contrast, Ternary Content Addressable Memory (TCAM)-based solutions are more viable for matching high-speed line rates while making software design fairly simple. A TCAM finds a matched rule in an $O(1)$ clock cycle and, therefore, offers the highest possible lookup/matching performance. However, despite its superior performance, it is still a challenge for a TCAM-based solution to match OC768 line rate or higher. Even the high-end solutions based on the state-of-the-art technologies, such as Cypress Ayama 20K [20], cannot keep up with OC768 line rate. Note that the I/O interface (2xLA-1) for Cypress Ayama 20K can afford at most 100 million 144-bit search key per second. Moreover, these solutions are generally not scalable and expensive, given that the memory speed improves by only 7 percent each year while the optical wire speed is doubling every 9-12 months [17].

In this paper, we develop a power efficient and scalable solution that can match OC768 and higher line rates, based on low-cost, low-speed TCAM technologies. Instead of striving to reduce the access latency for a single TCAM, we exploit Chip-Level-Parallelism (CLP) to achieve higher and scalable throughput. However, a naive approach is costly, given that TCAM is an expensive commodity, i.e., duplicating the databases to a set of uncoordinated TCAM chips. In

TABLE 1
An Example of Rule Table with 5-Tuple Rules

	Src IP	Dst IP	Src Port	Dst Port	Prot		Action
L1	1.1.x.x	2.2.2.x	any	any	6	→	AF
L2	x.x.x.x	2.2.x.x	any	256-512	6	→	BF
L3	3.3.x.x	x.x.x.x	>1023	512-1024	11	→	EF
L4	x.x.x.x	4.4.4.x	5000-6000	>1023	any	→	Accepted
L5	x.x.x.x	x.x.x.x	<1023	any	any	→	Discard
…	……	……	……	……	……	→	……

BF: Best effort Forwarding AF: Assured Forwarding EF: Expedited Forwarding

one of our previous works [16], it was demonstrated that, by making use of the structure of IPv4 route prefixes, a multi-TCAM solution that exploits CLP can actually achieve high throughput performance gain in supporting LPM with low memory cost.

Another important benefit of using TCAM CLP is its ability to effectively solve the range matching problem. Spitznagel et al. [10] reported that today's real-world Policy Filtering tables involve significant percentages of rules with ranges. Supporting rules with ranges or range matching without exquisite measures in TCAM will lead to very low TCAM storage efficiency, e.g., 16 percent as reported in [10]. Spitznagel et al. [10] proposed an extended-TCAM scheme to improve the TCAM storage efficiency, in which TCAM hierarchy and circuits for range comparisons are introduced. Another widely adopted solution to deal with range matching is to do a range preprocessing/encoding by mapping ranges to a short sequence of encoded bits, known as bit-mapping [11]. The application of the bit-map-based range encoding for packet classification using a TCAM was reported in [11], [12], [13], [14], [15]. A key challenge for range encoding is the need to encode multiple fields in a search key extracted from the packet to be classified at wire speed. To achieve high speed search key encoding, parallel search key field encoding was proposed in [11], [13], which, however, assume the availability of multiple processors and multiple memories for the encoding. To ensure the applicability of the range encoding scheme to any commercial network processors and TCAM coprocessors, the authors of this paper proposed using TCAM itself for sequential range encoding [15], which, however, reduces the TCAM throughput performance. Using TCAM CLP for range encoding provides a natural solution which solves the performance issue encountered in [15].

However, extending the idea in [16] to allow TCAM CLP for general policy filtering is a nontrivial task for the following two reasons: 1) The structure of a general policy rule, such as a 5-tuple rule, is much more complex than that of a route and it does not follow a simple structure like a prefix and 2) it requires matches on multiple dimensions (i.e., packet fields) with different matching conditions, such as prefix, range, and exact matches. Moreover, the search key is much longer than that for LPM. In this paper, we propose an efficient TCAM CLP scheme, called Distributed Parallel Packet Classification with Range Encoding (DPPC-RE), for the typical 5-tuple policy filtering. First, a rule database partitioning algorithm is designed to allow different partitioned rule groups to be distributed to different TCAMs with minimum redundancy. Then, a greedy heuristic algorithm is proposed to evenly balance the traffic load and storage demand among all the TCAMs. On the basis of these algorithms and combined with the range encoding ideas in [15], a fully adaptive algorithm is proposed to deal with range encoding and load balancing simultaneously. A thorough theoretical worst-case analysis of throughput, processing delay, and power consumption, as well as the experimental results, shows that the proposed solution can achieve scalable throughput performance matching OC768 line rate, with small TCAM storage overhead and bounded worst-case delay performance.

2 Terminology

Rules. A rule table or policy filtering table includes a set of *match conditions* and their corresponding *actions*. We consider the typical 104-bit 5-tuple match conditions, i.e., (SIP(1-32), DIP(1-32), SPORT(1-16), DPORT (1-16), and PROT(1-8)[3]), where SIP, DIP, SPORT, DPORT, and PROT represent source IP address, destination IP address, source port, destination port, and protocol number, respectively. DIP and SIP require prefix matching, SPORT and DPORT generally require range matching, and PROT requires exact matching. Except for fields with range matching, any other field in a match condition can be expressed using a single string of ternary bits, i.e., 0, 1, or "don't care"*. Table 1 gives an example of a typical 5-tuple rule table, here "x" = "********", a wildcard byte.

Rule Entry. TCAMs are organized in slots with fixed size (e.g., 64 or 72); each rule entry takes one or more slots depending on its size. Fig. 1 shows the implementation of rules L_1 and L_2 in a TCAM with 64-bit slots. Rule L_1 has no range in any of its fields and, hence, it takes two slots with 24 free bits in the second slot. Each such rule in the TCAM takes the minimum number of slots and is defined as a *rule entry*. L_2 has a range {256-512} in its destination port field. This range cannot be directly expressed as a string of ternary bits and must be partitioned into two subranges: {256-511} and {512}, expressed as: 0000 0001 **** **** and 0000 0010 0000 0000. Such a range that must be expressed by more than one ternary bit string is defined as the *nontrivial range*. Hence, L_2 takes four slots (slots 3, 4, 5, and 6) or two rule entries in the TCAM. In general, if ranges in the SPORT and DPORT fields in a match condition take n and m ternary strings, respectively, the match condition takes

3. The bits in the field are ordered with the first bit (MSB) lies in the leftmost position.

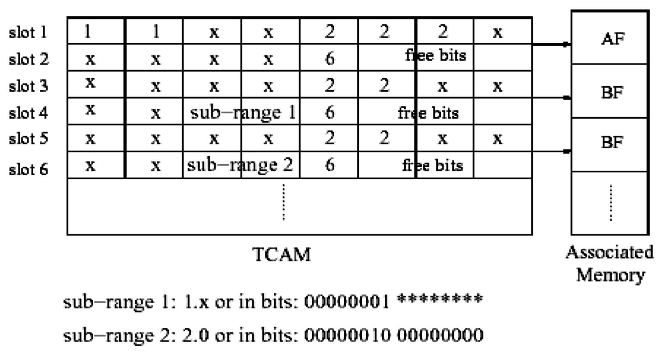

slot 1
slot 2
slot 3
slot 4
slot 5
slot 6
1 1 x x 2 2 2 x
x x x x 6 free bits
x x x x 2 2 x x
x x sub–range 1 6 free bits
x x x x 2 2 x x
x x sub–range 2 6 free bits
TCAM
AF
BF
BF
Associated Memory
sub–range 1: 1.x or in bits: 00000001 ********
sub–range 2: 2.0 or in bits: 00000010 00000000

Fig. 1. Rules in a TCAM. Range {256-512} is split into two subranges, {256-511} and {512}, and implemented as subrange 1 and subrange 2. The other numbers represent the actual byte values.

$n \times m$ TCAM rule entries. This multiplicative expansion of the TCAM usage to support range matching is the root that causes low TCAM storage efficiency.

Range Encoding. An efficient solution to deal with range matching is to map a range to a short sequence of encoded bits, known as range encoding [15]. After range encoding, a rule with encoded ranges only takes one rule entry, thus improving TCAM storage efficiency.

N_0: rule table size, or number of rules in a rule table. N: number of TCAM entries required to accommodate a rule table without range encoding. N_e: number of TCAM entries required to accommodate a rule table with range encoding.

Search Key. A search key is a 104-bit string composed of a 5-tuple. For example, <1.1.1.1, 2.2.2.2, 1028, 30065, 11> is a 5-tuple search key. In general, a search key is extracted from the packet header and passed to a packet classifier to match against a 5-tuple rule set.

Matching. In the context of a TCAM-based solution as is the case in this paper, matching refers to ternary matching in the following sense: A search key is said to match a particular match condition if, for each and every corresponding bit position in both the search key and the match condition, either of the following two conditions is met: 1) the bit values are identical or 2) the bit in the match condition is "don't care" or *.

So far, we have defined the basic terminologies for rule matching. Now, we establish some important concepts upon which the distributed TCAM is developed.

ID. The idea of the proposed distributed TCAM is to make use of a small number of bits extracted from certain bit positions in the search key and match condition as IDs to 1) divide rules into groups, which are mapped to different TCAMs, and 2) direct a search key to a specific TCAM for Rule-Matching. In this paper, we use P number of bits picked from given bit positions in the DIP, SIP, and/or PROT fields of a match condition as the rule ID, denoted Rule-ID, for the match condition and use P number of bits extracted from the corresponding search key positions as the key ID, denoted Key-ID, for the search key. For example, suppose $P = 4$ and they are extracted from SIP(1), DIP(7), DIP(16), and PROT(8). Then, the rule-ID for the match condition <1.1.x.x, 2.x.x.x, x, x, 6> is "01*0" and the Key-ID for the search key <1.1.1.1, 2.2.2.2, 1028, 34556, 17> is "0101."

ID Groups. We define all the match conditions having the same Rule-ID as a *Rule-ID group*. Since a Rule-ID is composed of P ternary bits, the match conditions or rules are classified into 3^P Rule-ID groups. If "*" is replaced with "2," we get a ternary value for the Rule-ID which uniquely identifies the Rule-ID group (note that the numerical value for different Rule-IDs are different). Let RID_j be the Rule-ID with value j and RG_j represent the Rule-ID group with Rule-ID value j. For example, for $P = 4$, the Rule-ID group with Rule-ID "00*1" is RG_7 since the Rule-ID value $j = \{0021\}_3 = 7$. Accordingly, we define the set of all the Rule-ID groups with their Rule-IDs matching a given Key-ID as a *Key-ID group*. Since each Key-ID is a binary value, we use this value to uniquely identify this Key-ID group. In parallel to the definitions for Rule-ID, we define Key-ID KID_i with value i as a *Key-ID group* KG_i. We have a total number of 2^P *Key-ID groups*. With the above definitions, we have $KG_i = \bigcup_{RID_j \text{ match } KID_i} RG_j$. For instance, suppose $P = 2$ and SIP(16) and DIP(16) are chosen as ID bits. Then, the Rule IDs of the five rules introduced in Table 1 are "10," "*0," "1*," "*0," and "**," respectively. Table 2 depicts the Key-ID group and Rule-ID group hierarchy of rule set.

Distributed Storage Expansion Ratio. Since Key-ID groups may overlap with one another, we have: $\sum_i |KG_i| \geq |\bigcup_i KG_i|$, where $|A|$ represents the number of elements in set A. In other words, using Key-ID to partition rules and distribute them to different TCAM introduces redundancy. To formally characterize this effect, we further define *Distributed storage Expansion Ratio (DER)* as $DER = D(N,K)/N$, where $D(N,K)$ represents the total number of TCAM entries required to accommodate N rules when rules are distributed to K different TCAMs. Here, DER characterizes the redundancy introduced by the distributed storage of rules with or without range encoding.

Throughput and Traffic Intensity. In this paper, we use throughput, traffic intensity, and throughput ratio as performance measures of the proposed solution. Two measurements can be adopted to quantify *Throughput*. One is the number of packets classified per unit time (denoted *packet per second* or pps), which denotes the ability of the classification engine to handle packets, and the other is the number of packets classified per clock cycle (denoted as *packets per cycle* or ppc), which actually denotes the level of parallelism of the classification engine. It is an important measure of the processing power of the proposed solution.

TABLE 2
ID Groups Hierarchy of the Five Rules Introduced in Table 1

Key-ID Groups	KG_2 ("10")				
Rule-ID Groups	RG_3 ("10")	RG_6 ("*0")		RG_5 ("1*")	RG_8 ("**")
Rule No.	**L1**	**L2**	**L4**	**L3**	**L5**

Traffic intensity is used to characterize the workload in the system. As the design is targeted at OC768 line rate, we define traffic intensity as the ratio between the actual traffic load and the worst-case traffic load at OC768 line rate, i.e., 125 Mpps. *Throughput ratio* is defined as the ratio between *Throughput* and the worst-case traffic load at OC768 line rate. The following theorem states under what conditions the proposed solution maintains the original packet ordering:

Theorem 1. *The original packet ordering for any given application flow is maintained if packets with the same Key-ID are processed in order.*

Proof. First, note that packet ordering should be maintained only for packets belonging to the same application flow and an application flow is, in general, identified by the 5-tuple. Second, note that packets from a given application flow must have the same Key-ID by definition. Hence, the original packet ordering for any given application flow is maintained if packets with the same Key-ID are processed in order. □

3 Algorithms and Solutions

The key problems we aim to solve are 1) how to make use of CLP to achieve high packet classification performance with minimum cost and 2) how to combine CLP with TCAM range encoding schemes to improve the TCAM storage efficiency (consequently controlling the cost and power consumption). A scheme called Distributed Parallel Packet Classification with Range Encoding (DPPC-RE) is proposed. The idea of DPPC is the following: First, by appropriately selecting the ID bits, a large rule table is partitioned into several Key-ID groups of similar sizes. Second, by applying certain load balancing and storage-balancing heuristics, the rules (Key-ID groups) are distributed evenly to several TCAM chips. As a result, multiple packet classifications corresponding to different Key-ID groups can be performed simultaneously, which significantly improves throughput performance without incurring much additional cost. The idea of RE is to encode the range fields of the rules and the corresponding fields in a search key into bit-vectors, respectively. In this way, the number of ternary strings (or TCAM entries) required to express a rule with nontrivial ranges can be significantly reduced (e.g., to only one string), improving TCAM storage efficiency. In DPPC-RE, the TCAM chips that are used to perform Rule-Matching are also used to perform search key encoding. This not only offers a natural way for parallel search key encoding, but also makes it possible to develop efficient load balancing schemes (which will be revealed soon in Section 3.4), making DPPC-RE a practical solution. In what follows, we introduce DPPC-RE in detail.

3.1 ID-Bit Selection

The objective of ID-bit selection is to minimize the number of redundant rules (introduced due to the overlapping among Key-ID groups) and to balance the size of the Key-ID groups (large discrepancy of the Key-ID group sizes may result in low TCAM storage utilization). A brute-force approach to solve the above optimization problem would be to traverse all of the P-bit combination out of W-bit rules

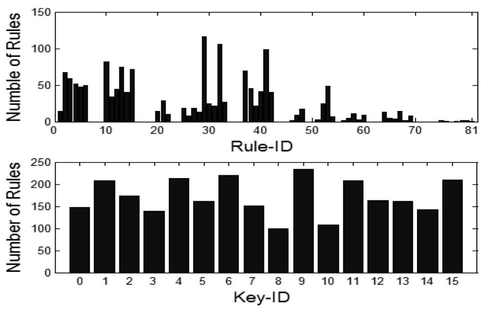


Fig. 2. ID-bit selection result of rule database set #5.

to get the best solution. However, since the value of W is relatively large (104 bits for the typical 5-tuple rules), the complexity is generally too high to do so. Hence, we introduce a series of empirical rules based on the analysis of the five real-world databases [18] that are used throughout the rest of the paper to simplify the computation as follows:

1. Since the fields DPORT and SPORT in a rule may have nontrivial ranges which need to be encoded, we choose not to take these two fields into account for ID-bit selection.
2. According to the analysis of several real-world rule databases [18], over 70 percent of the rules are with nonwildcarded PROT field, and over 95 percent of these nonwildcarded PROT fields are either TCP(6) or UDP(17) (approximately 50 percent are TCP). Hence, one may select either the eighth or the fifth bit (TCP and UDP have different values at these two bit positions) of the PROT field as one of the ID bits. All the rest of the bits in the PROT field have a fixed one-to-one mapping relationship with the eighth or fifth bits and do not lead to any new information about the PROT.
3. Note that rules with wildcard(s) in their Rule-IDs are actually those incurring redundant storage. The more wildcards a rule has in its Rule-ID, the more Key-ID groups it belongs to and, consequently, the more redundant storage it incurs. In the five real-world rule databases, there are over 92 percent of the rules whose DIP values are prefixes no longer than 25 bits and over 90 percent of the rules whose SIP values are prefixes no longer than 25 bits. So, we choose not to use the last 7 bits (i.e., the 26th to 32nd bits) of these two fields since they are wildcards in most cases.

Based on these three empirical rules, the traversal is simplified as follows: Choose an optimal $(P-1)$-bit combination (we can afford to perform a brute-force search since only 50 out of 104 bits are considered) out of 50 bits of DIP and SIP fields (DIP(1-25), SIP(1-25)), and then combine these $(P-1)$ bits with PROT(8) or PROT(5) to form the P-bit ID. Fig. 2 shows an example of the ID-bit selection for Database #5 [18] (the largest database, with 1,550 rules, among the five databases studied in this paper). We use an equally weighted sum of two objectives, i.e., the minimization of the variance among the sizes of the Key-ID groups and the total number of redundant rules, to find the 4-bit combination: PROT(5), DIP(1), DIP(21), and SIP(4).[4] We find that, although the sizes of the Rule-ID groups are unbalanced, the sizes of the Key-ID groups are quite similar, which allows memory-efficient schemes to be developed for the distribution of rules to TCAMs.

3.2 Distributed Table Construction

The next step is to evenly distribute the Key-ID groups to K TCAM chips and to balance the classification load among them. For clarity, we first describe the mathematical model of the distributed table construction problem as follows: Let S be the set of all the rules, Q_k be the set of the Key-ID groups placed in TCAM #k $(k = 1, 2, \dots, K)$, $W[j], j = 1, \dots, 2^P$ be the frequency of KID_j appearances in the search keys, indicating the Rule-Matching load ratio of Key-ID group KG_j, $RM[k]$ be the Rule-Matching load ratio that is assigned to TCAM #k, namely, $RM[k] := \sum_{j \in Q_k} W[j]$, $G[k]$ be the number of rules distributed to TCAM #k, namely, $G[k] := |\bigcup_{KG_i \in Q_K} KG_i|$, and C_t be the maximum number of rules each TCAM can contain (capacity tolerance). The optimization problem for distributed table construction is given by: To find a K-division $\{Q_k, k = 1, \dots, K\}$ of the Key-ID groups that:

$$Minimize : Max_{k=1,\dots,K} RM[k] \text{ and } Max_{k=1,\dots,K} G[k];$$
$$Subject\ To : Q_k \in S, \bigcup_{k=1,\dots,K} Q_k = S,$$
$$\text{and } Max_{k=1,\dots,K} G[k] \le C_t.$$

Consider each Key-ID group as an object and each TCAM chip as a knapsack. We find that the problem is actually a variance of the *Weighted-Knapsack* problem, which can be proven to be NP-hard (please refer to the appendix in [16] for a detailed discussion). Note that the problem has multiple objectives, which cannot be handled by conventional greedy methods. In what follows, by partitioning the original problem into two cases and defining the corresponding priority rule of the two objectives under certain considerations in each case, we turn the original multiple-objective optimization problem into a single-objective one. Then, we further develop a systemic heuristic to solve the problem.

3.2.1 Distributed Table Construction Scheme (DTCS)

Parameter Initiation (PI). The number of TCAM chips, K, may be set at an integer value, which should be no less than $\lceil Thr_{req}/Thr_{TCAM} \rceil$, where Thr_{req} is the throughput objective to be achieved and Thr_{TCAM} is the throughput a single TCAM chip can provide. This will guarantee that the requested throughput performance is met. Also note that a larger K value may result in higher expected system performance, but also in higher expected cost (e.g., caused by DER). Import all ID bits selection and traffic load information. For k from 1 to K, let $Q_k = \Phi$, $\text{RM}[k] = 0$, and $\text{G}[k] = 0$.

Note that, if the two objectives are consistent with each other, the optimization problem may actually be transformed into a single objective problem, which is much easier to solve. Therefore, for generality, we consider in the following discussion that the two objectives compete with each other, i.e., the Key-ID groups with relatively large traffic load ratio tend to have relatively small capacity requirement. On the other hand, as will be revealed soon in Section 3.4 (according to the FA algorithm in that subsection), whether the condition $Max_{k=1,\dots,K} RM[k] \le 2/\text{K}$ can be satisfied or not determines whether or not the classification traffic load can be perfectly balanced. Thus, we develop the Capacity First Algorithm (CFA) and the Load First Algorithm (LFA) corresponding to the two cases, respectively.

```
i) sort $\{i, i=1,2,\dots,2^P\}$ in decreasing order of $|KG_i|$ and
   record the result as $\{Kid[1],\dots,Kid[2^P]\}$
ii) for $i$ from 1 to $2^P$ do
      sort $\{k, k=1,\dots,K\}$ in increasing order of $G[k]$ and
      record as $\{Sc[1], Sc[2],\dots,Sc[K]\}$ ;
      for $k$ from 1 to $K$ do
        if $RM[Sc[k]]+W[Kid[i]] \le L_t$ then
          $Q_{Sc[k]} = Q_{Sc[k]} \cup KG_{Kid[i]}$ ;
          $G[Sc[k]] = |Q_{Sc[k]}|$;
        $RM[Sc[k]] = RM[Sc[k]] + W[Kid[i]]$;
iii) output $\{Q_k, k=1,\dots,K\}$ and $\{RM[k], k=1,\dots K\}$.
```

Fig. 3. The pseudocode for CFA.

3.2.2 The CFA Phase

We first consider *minimizing* $Max_{k=1,\dots,K} G[k]$ the primary objective, while *minimizing* $Max_{k=1,\dots,K} RM[k]$ serves as a secondary objective by introducing a constraint set $RM[k] \le L_t, k = 1, \dots, K$. In the proposed scheme, variable L_t is initially set to be the minimum threshold, i.e., 1/K, and would be loosened when no feasible solution can be found in CFA. Suppose that CFA terminates without a feasible solution in the current loop; in other words, no matter which TCAM a certain Key-ID groups, e.g., KG_i, is allocated to, constraint $RM[k] + W[i] \le L_t$ does not hold. In this case, we define $\Delta L := Min_{k=1,\dots,K} RM[k] + W[i] - L_t$ and just loosen L_t by ΔL. Namely, we always keep L_t the possible minimum value for producing a feasible solution. On the other hand, in CFA, the Key-ID groups with relatively more rules will be distributed first and the current Key-ID group will be assigned to the TCAM with the least number of rules under the load constraint. The pseudocode of the CFA is shown in Fig. 3.

3.2.3 The LFA Phase

After running CFA, if $Max_{k=1,\dots,K} RM[k] < 2/K$ holds, the result from CFA is final and the process of DTCS terminates. Otherwise, we turn to the phase of LFA in which we consider *minimizing* $Max_{k=1,\dots,K} RM[k]$ as the primary objective, while the objective of *minimizing* $Max_{k=1,\dots,K} G[k]$ is relaxed to a constraint, i.e., $G[k] \le C_t, k = 1, \dots, K$. We explore the trade-off between the two objectives by adjusting C_t. In practice, we set $C_t = Max_{k=1,\dots,K} G[k]$, which is obtained in the CFA phase initially, and loosen it adaptively until a feasible solution is found in LFA. More specifically, we loosen C_t in the same manner as L_t in CFA to always keep it the minimum value for a feasible solution. In LFA, we allocate K Key-ID groups at a time and call such a process *a round*.

4. The leftmost bit is the least significant bit.

```
i) sort $\{i, i=1,\ldots,2^P\}$ in decreasing order of $W[i]$ and record the
   result as $\{Kid[1],\ldots,Kid[2^P]\}$ ; Let $N = N' = \lfloor 2^P / K \rfloor$;
   if $N' \times K < 2^P$ then
      $N = N'+1$;
      for $i$ from $2^P+1$ to $(N'+1)K$
         define $W[Kid[i]] = 0, |KG_{Kid[i]}| = 0$ |;
ii) for $i$ from 1 to $N$ do
      sort $\{k, k=1,\ldots,K\}$ in increasing order of $RM[k]$ and record
      as $\{Sc[1],\ldots,Sc[K]\}$;
      Take $\{Kid[(i-1)K+1],\ldots,Kid[(i-1)K+K]\}$ to be allocated in the
      current round; pointer = $(i-1)K+1$;
      for $k$ from 1 to $K$ do
         if $|Q_{Sc[k]} \cup KG_{Kid[pointer]}| \le C_t$ then
            $Q_{Sc[k]} = Q_{Sc[k]} \cup KG_{Kid[pointer]}$;
            $G[Sc[k]] = |Q_{Sc[k]}|$;
            $RM[Sc[k]] = RM[Sc[k]] + W[Kid[pointer]]$; pointer++;
iii) output $\{Q_k, k=1,\ldots,K\}$ and $\{RM[k], k=1,\ldots,K\}$ .
```

Fig. 4. The pseudocode for LFA.

Note that K may not exactly divide 2^P, so there may be $n(n < K)$ Key-ID Groups left after a certain number of rounds. In this case, we still call these residual Key-ID Groups a round by adding $(K - n)$ *virtual Key-ID Groups* whose $W[i]$ and $|KG_i|$ are all 0. The Key-ID groups with relatively larger traffic load ratio will be assigned to TCAM first. Each Key-ID group will be assigned to the TCAM chips with relatively lower load. The pseudocode of the LFA is shown in Fig. 4.

Based on these two algorithms, the flow of the distributed algorithm is depicted in Fig. 5 and summarized as the following: At the beginning, CFA is run. If no feasible solution is found in a loop, the constraint L_t is loosened iteratively. Whenever a feasible solution is found, the inequality $L_t \le 2/K$ is checked: If it holds, return final results; otherwise, run LFA to get the final result. Theoretical analysis of DTCS will be presented in Section 5. Here, we again use the rule database set #5 as an example (more results for the *DER*s on the other four rule databases will be discussed in Section 5.6). Suppose that the traffic load distribution among the Key-ID groups is as depicted in Fig. 6, which is selected intentionally to have a large variance to create a difficult case for load balancing. Note that the ID-bits are PROT(5), DIP(1), DIP(21), and SIP(4) as obtained in the last subsection and constraint $C_t = 560$. For $K = 5$, the final result is depicted in Table 3 (the maximum traffic load ratio is $24.4\% < 2/K = 40\%$). We note that the numbers of rules assigned to different TCAMs are very close to one another. The *DER* is $C_t \times K/N_0 = 1.80$, meaning that only about 80 percent more TCAM entries are required (note that $K^*100\%$ are required in the case when the rule table is duplicated and assigned to each TCAM). As we shall see shortly, using the load balancing scheme proposed in Section 3.4, this kind of traffic distribution can be perfectly balanced.

3.3 Solutions for Range Matching

Range matching is a critical issue for effective use of TCAM for policy filtering. The real world databases in [10] showed that TCAM storage efficiency can be as low as 16 percent due to the existence of a large number of rules with ranges. We apply our earlier proposed Dynamic Range Encoding

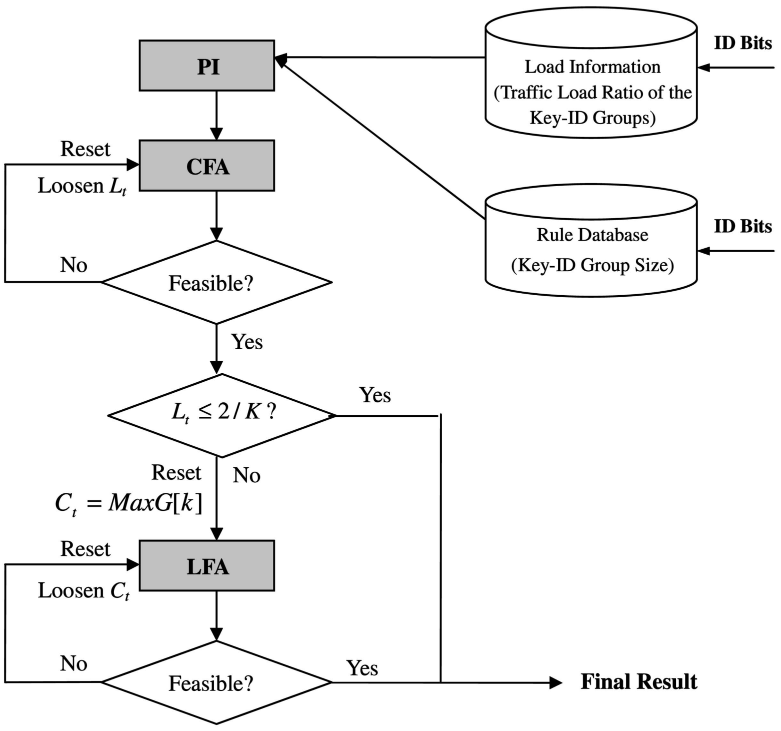


Fig. 5. Distributed table construction scheme.

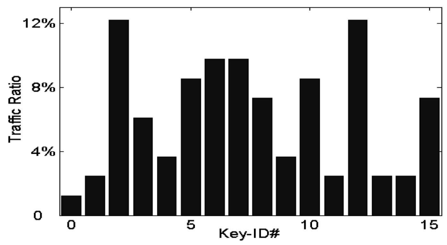


Fig. 6. Traffic load distribution among the Key-ID (Key-ID groups).

Scheme (DRES) [15] to distributed TCAMs to improve the TCAM storage efficiency. DRES [15] makes use of the free bits in each rule entry to encode a subset of ranges selected from any rule field with ranges. An encoded range is mapped to a code vector implemented using the free bits and the corresponding field is wild carded. Hence, a rule with encoded ranges can be implemented in one rule entry, reducing the TCAM storage usage. To match an encoded rule, a search key is preprocessed to generate an encoded search key. This preprocess is called Key-Encoding. Accordingly, the process in a TCAM with range encoding includes two steps: key-encoding and rule-matching. DRES uses the TCAM coprocessor itself for key-encoding to achieve wire speed performance. If the encoded ranges come from S fields, S separate range tables are needed for key-encoding. The S range tables as well as the rule table can be allocated in the same or different TCAMs. The key-encoding involves S fields matching against the corresponding S range tables to get an encoded search key. Then, the encoded search key is matched against the rule table to get the final result. In summary, a rule match with range encoding requires S range table lookups and one rule table lookup.

Similarly to DRES, suppose that low-end TCAM chips with 133 MHz clock rate and 64-bit slot size are adopted. For typical 104-bit 5-tuple rules, ranges only appear in the source and destination port fields and, hence, only two range tables are needed. For a TCAM with 64-bit slot size, each rule takes two slots and leaves 24 free bits for range encoding. Each Rule-Matching takes two TCAM matches (each slot takes one). In each range table, a range takes one slot and a range matching incurs one TCAM lookup. In summary, there are a total number of four TCAM matches per search key search (two for key encoding and another two for rule-matching). With a 133 MHz TCAM at 133 million lookups per second, DRES can barely support OC192 wire speed performance. In this paper, the distributed TCAM scheme that exploits CLP to increase the TCAM lookup performance is proposed to support line rates higher than OC192, e.g., OC768. Moreover, a slower TCAM technique may be adopted in the distributed TCAM scheme, e.g., the 100 MHz TCAM chips, and, therefore, result in saving in terms of both cost and power consumption. In Section 4, we will present the details on how to incorporate DRES into the proposed distributed solution.

3.4 Adaptive Load-Balancing Scheme

Note that the DPPC formulation is static in the sense that, once the Key-ID groups are populated in different TCAMs, the performance is pretty much subject to traffic pattern changes. The inclusion of range encoding provides us with a very efficient way to dynamically balance the traffic in response to traffic pattern changes. The key idea is to duplicate range encoding tables to all the TCAMs and, hence, allow a key-encoding to be performed using any one of the TCAMs to dynamically balance the load. Since the size of the range tables is small, e.g., no more than 15 entries for all five real-world databases, duplicating range tables to all the TCAMs does not impose a distinct overhead. Again, we first define some mathematical terms. Let $D[k]$ be the overall traffic load ratio assigned to TCAM #k $(k = 1, 2, \dots, K)$, which includes two parts, i.e., the key-encoding traffic load ratio and the rule-matching traffic load ratio, with each contributing 50 percent of the load, according to Section 3.3. Let $KE[k]$ and $RM[k]$ be the key-encoding and rule-matching traffic ratio allocated to TCAM #k, $(k = 1, 2, \dots, K)$, respectively. Note that $RM[k]$ is determined by the DTCS process (refer to Section 3.2). Let $A[i, k]$, $A[i, k] \geq 0, i, k = 1, \dots, K, \sum_i A[i, k] = 1$, be the *Adjustment Factor Matrix*, where $A[i, k]$ is defined as the percentage (ratio) of the key-encoding tasks allocated to TCAM #i, for the corresponding Rule-Matching tasks which are performed in TCAM #k. Then, the dynamic load balancing problem is formulated as follows:

Decide $A[i, k]$, $i, k = 1, \dots, K$, when ***Minimize***:
$Max_{k=1,\dots,K} D[k]$.
Subject to: $D[k] = 0.5 \times KE[k] + 0.5 \times RM[k]$, $k = 1, \dots, K$,
and $KE[i] = \sum_{k=1,\dots,K} A[i, k] \times RM[k]$, $i = 1, \dots, K$.

The following algorithm is proposed to solve the above problem.

3.4.1 *Full Adaptation (FA)*

The idea of FA is to use a counter to keep track of the current number of backlogged tasks in the buffer at each

TABLE 3
Distributed Table Construction Result, Derived from CFA at $K = 5$

TCAM	Key-ID Groups (Table Contents)				Number of Rule-ID Groups	Number of Rules	Traffic Load Ratio%
#1	11(1011)	2(0010)	0(0000)		36	478	15.9
#2	8(1000)	7(0111)	4(0100)		40	439	20.7
#3	15(1111)	10(1010)	14(1110)		40	361	18.3
#4	9(1001)	3(0011)	13(1101)	12(1100)	36	556	24.4
#5	5(0101)	6(0110)	1(0001)		36	494	20.7
Distributed storage Expansion Ratio (*DER*)						560*5/1550=2800/1550=1.80	

No iteration is needed.

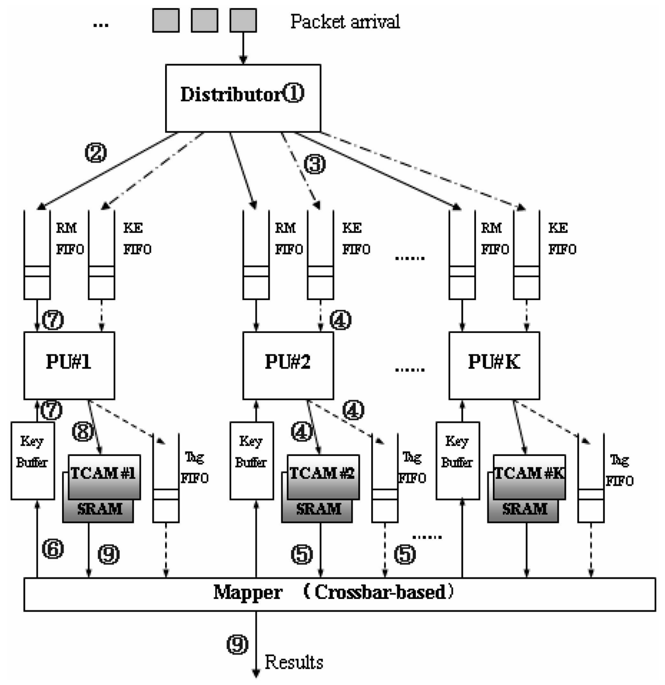


Fig. 7. DPPC-RE architecture.

TCAM chip. Whenever a packet arrives, the corresponding key-encoding task is assigned to the TCAM which has the smallest counter value. In this case, the values of $A[k, i]$ are not fixed (dynamic). The expression of $D[k]$ is given by:

$$D[k] = 0.5 \times RM[k] + 0.5 \times (A[k, 1] \times RM[1]] + \dots + A[k, K] \times RM[K]).$$

Note that $0 \leq A[k, i] \leq 1$, $i = 1, \dots, K$, and we have $0.5 \times RM[k] \leq D[k] \leq 1$, $k = 1, \dots, K$. Taking $A[i, k]$ as tunable parameters, it is clear that the equations:

$$1/K = D[k] = 0.5 \times RM[k] + 0.5 \times (A[k, 1] \times RM[1] + \dots + A[k, K] \times RM[K]), k = 1, \dots, K,$$

must have feasible solutions when $0.5 \times RM[k] \leq 1/K$, i.e., $RM[k] \leq 2/K$, $k = 1, \dots, K$.

This means that if the conditions $RM[k] \leq 2/K$, $k = 1, \dots, K$, hold, the overall traffic load ratio can be perfectly balanced (i.e., the objective value is $1/K$) in the presence of traffic pattern changes. Further discussions on the performance of FA are presented in Section 5.

4 Implementation of the DPPC-RE Scheme

The detailed implementation of the DPPC-RE mechanism is depicted in Fig. 7. Besides the TCAM chips and the associated SRAMs to accommodate the match conditions and the associated actions, three major additional components are included in cooperating with the TCAM chips, i.e., a distributor, a set of processing units (PUs), and a mapper. Some associated small buffer queues are used as well. Now, we describe the components in detail.

4.1 Distributor

This component is actually a scheduler. It partitions the traffic to be loaded among the TCAM chips. More specifically, it performs three major tasks. First, it extracts the key-ID from the 5-tuple search key received from a network processing unit (NPU). The Key-ID is used as an identifier to dispatch the rule-matching keys to the associated TCAM. The 5-tuple key is pushed into the rule-matching FIFO (RM FIFO) queue of the corresponding TCAM (solid arrows in Fig. 7). Second, the distributor distributes the key-encoding traffic among the TCAM chips, according to the FA algorithm. The corresponding information, i.e., the SPORT and DPORT, is pushed into the KE FIFO of the TCAM selected (dashed arrows in Fig. 7). Third, the distributor maintains K serial numbers (S/N) or S/N counters, one for each TCAM. An S/N is used to identify each incoming packet (or, more precisely, each incoming 5-tuple). Whenever a packet arrives, the distributor adds "1" (cyclical with modulus equal to the RM FIFO depth) to the S/N counter for the corresponding TCAM the packet is mapped to. A tag is defined as the combination of an S/N and a TCAM number (CAMID). This tag is used to uniquely identify a packet and its associated rule-matching TCAM. The format of the tag is depicted in Fig. 8a. As we shall explain shortly, the tag is used by the mapper to return the key-encoding results back to the correct TCAM and to allow the PU for that TCAM to establish the association of these results with the corresponding 5-tuple key in the rule-matching queue.

4.2 RM FIFO, KE FIFO, Key Buffer, and Tag FIFO

A Rule-Matching FIFO (RM FIFO) is a small FIFO queue where the information for rule-matching of the incoming packets is held. The format of each unit in the RM FIFO is given in Fig. 8b (the numbers in the brackets indicate the number of memory bits needed for the fields). A key-encoding FIFO (KE FIFO) is a small FIFO queue where the information used for key-encoding is held. The format of

S/N(5)	CAMID(3)

(a)

PROT(8)	DIP(32)	SIP(32)	DPORT(16)	SPORT(16)	Tag(8)

(b)

DPORT(16)	SPROT(16)	Tag(8)

(c)

Valid(1)	DPK(8)	SPK(8)

(d)

Fig. 8. Format of tag, RM FIFO, KE FIFO, and Key Buffer. (a) Tag format. (b) RM FIFO format. (c) KE FIFO format. (d) Key Buffer format. SPK/DPK: encoded source/destination port key.

each unit in the KE FIFO is given in Fig. 8c. Differently from the RM and KE FIFOs, a key buffer is not a FIFO queue, but a fast register file accessed using an S/N as the address. It is where the results of key-encoding (encoded bit vectors of the range fields) are held. The size of a key buffer equals the size of the corresponding RM FIFO, with one unit in the key buffer corresponding to one unit in the RM FIFO. The format of each unit is given in Fig. 8d. The *Valid* bit is used to indicate whether the content is available and up-to-date. Note that the tags of the key cannot be passed through TCAM chips during the matching operations. Hence, a tag FIFO is designed for each TCAM chip to keep the tag information when the associated keys are being matched.

4.3 The Processing Unit

Each TCAM is associated with a processing unit (PU). A PU performs the following two functions: First, it schedules the rule-matching and key-encoding tasks assigned to the corresponding TCAM, aiming at maximizing the utilization of the corresponding TCAM. Second, it ensures that the results of the incoming packets assigned to this TCAM are returned in order. In what follows, we elaborate on these two functions.

4.3.1 Scheduling between Rule-Matching and Key-Encoding Tasks

Note that, for any given packet, the rule-matching operation cannot take place until the key-encoding results are returned. Hence, it is apparent that the units in an RM FIFO would wait for a longer time than the units in a KE FIFO. For this reason, rule-matching tasks should be assigned higher priority than key-encoding tasks. However, our analysis indicates that a strict-sense priority scheduler may lead to nondeterministically large processing delay. So, we introduce a weighted-round-robin scheme in the PU design. More specifically, each type of tasks gains higher priority in turn based on an asymmetrical round-robin mechanism. In other words, the key-encoding tasks will gain higher priority for one turn (one turn represents two TCAM accesses, for either a rule-matching operation or two successive key-encoding operations) after n turns with the higher priority assigned to rule-matching tasks. Here, n is defined as the *round-robin-ratio*. As we shall see in Section 5.2, this scheduling algorithm will ensure that the maximum processing delay experienced by each packet is upper bounded.

4.3.2 Ordered Processing

Apparently, the order of the returned results from a specific TCAM is determined by the processing order of the rule-matching operation. Since a rule-matching buffer is a FIFO queue, the results will be returned in the same order as the packet arrivals, although the key-encoding tasks of the packets may not be processed in their original sequence.[5] As a result, if the key-encoding result for a given rule-matching unit returns earlier than those units in front of it, this rule-matching unit cannot be executed. Specifically, the PU for a given TCAM maintains a pointer pointing to the position in the Key Buffer that contains the key-encoding result corresponding to the unit at the head of the RM FIFO. The value of the pointer equals the S/N of unit at the head of RM FIFO. In each TCAM cycle, PU queries the valid bit of the position that the pointer points to in the key buffer. If the bit is set, meaning that the key-encoding result is ready, and it is rule-matching's turn for execution, PU reads the key-encoding results out from the key buffer and the 5-tuple information out from the RM FIFO queue, and launches the rule-matching operation. Meanwhile, the valid-bit of the current unit in the key buffer is reset and the pointer is incremented by 1 in a cyclical fashion. Since the S/N for a packet in a specific TCAM is assigned cyclically by the distributor, the pointer is guaranteed to always point to the unit in the key buffer that corresponds to the head unit in the RM FIFO.

5. This is because the key encoding tasks whose rule-matching is processed in a specific TCAM may be assigned to different TCAMs to be processed based on the FA algorithms.

4.4 Mapper

The function of this component is to manage the result returning process of the TCAM chips. According to the processing flow of an operation, the mapper has to handle three types of results, i.e., the key-encoding phase-I results (for the SPORT field), the key-encoding-phase-II results (for the DPORT field), and the rule-matching results. The type of the result is encoded in the result itself. If the result from any TCAM is a rule-matching result, the mapper returns it to the NPU directly. If it is a key-encoding-phase-I result, the mapper stores it in a latch and waits for the Phase II result, which will come in the next cycle. If it is a key-encoding-phase II result, the mapper uses the tag information from the Tag FIFO to determine: 1) which key buffer (according to the CAMID segment) this result should be returned to and 2) which unit in the key buffer (according to the S/N segment) this result should be written into. Finally, the mapper combines the two results (of phases I and II) into one and returns it.

4.5 An Example of the Processing Flow

Suppose that the ID-bit selection is based on rule database #5 and the four ID-bits are PROT(4), DIP(1), DIP(21), and SIP(4). The distributed rule table is given in Table 2 (in Section 3.2). Given a packet $P0$ with 5-tuple, <166.111.140.1, 202.205.4.3, 15335, 80, 6>, the processing flow is the following (also shown in Fig. 7):

1. The 4-bit Key-ID "0010" is extracted by the distributor.
2. According to the distributed rule table given by Table 2, key-ID group "0010" is stored in TCAM#1 (i.e., with CAMID "001"). Suppose that the current S/N value of TCAM#1 is "00101(5)," then the CAMID "001" and the updated S/N "00110(5+1)" are combined into the tag with value "00110001." Then, the 5-tuple, together with the tag, is pushed into the RM FIFO of TCAM#1.
3. Suppose that the current queue sizes of the five KE FIFOs are 2, 0, 1, 1, and 1, respectively. According to the FA algorithm, the key-encoding operation of packet $P0$ is to be performed in TCAM#2. Then, the two range fields <15535, 80>, together with the tag,

are pushed into the KE FIFO associated with TCAM#2.

4. Suppose that now it is the turn for key-encoding or no rule-matching task is ready for execution, PU#2 pops off the head unit (<15535, 80>+Tag<00100110>) from the KE FIFO and sends them to TCAM#2 to perform the two range encodings successively. Meanwhile, the corresponding tag is pushed into the Tag FIFO.
5. When both results are received by the mapper, it combines them into one and pops the head unit from the tag FIFO.
6. The CAMID field "001" in the tag indicates the result should be sent back to the key buffer of TCAM#1, while the S/N field "00110" indicates that it should be stored in the sixth unit of the key buffer. Meanwhile, the corresponding valid bit is set.
7. Suppose that all the packets before packet $P0$ have been processed and $P0$ is now the head unit in the RM FIFO of TCAM#1. Note that packet $P0$ has S/N "00110". Hence, when it is the rule-matching's turn, PU#1 probes the valid bit of the sixth unit in the key buffer.
8. When PU#1 finds that the bit is set, it pops the head unit from the RM FIFO (the 5-tuple) and reads the contents out from the sixth unit of the key buffer (the encoded key of the two ranges) and then launches a rule-matching operation in TCAM#1. Meanwhile, the valid bit of the sixth unit in the key buffer is reset and the pointer of PU#1 is incremented by one and points to the seventh unit.
9. When the mapper receives the rule-matching result, it returns the result to the NPU, completing the whole process cycle for packet $P0$.

5 Performance Evaluation

5.1 Throughput

After a thorough investigation into the original problem, we find that it is very hard to provide precise theoretical analysis for the performance evaluation of DTCS (more precisely, LFA algorithm in DTCS). However, with some reasonable assumptions made, which are derived from numerous experiments and observations, we are able to evaluate the worst-case throughput performance of the proposed DTCS as the following:

Assumption 1. *Suppose that, for any KG_j, $j = 1, \ldots, 2^p$ and a given capacity constraint C_t, we have $|KG_j| < 0.5C_t$ and $W[j] < 2 \times (1/2^P) = 1/2^{P-1}$* **(A1)**.

This assumption is derived from the key observation that, although it is possible that the Key-ID group with a relatively large number of rules or heavy traffic load ratio may exist, it is unlikely to see one, in practice, which occupies more than half of the TCAM capacity or whose traffic load ratio exceeds more than twice the average traffic load ratio of each ID group.

Assumption 2. *Sorting $W[j], j = 1, \ldots, 2^P$ in decreasing order and denoting the result as $W'[j], j = 1, \ldots, 2^P$ and defining $W'[2^P + 1] \equiv 0$, we then have*

$$W'[1] \geq W'[2] \geq \ldots W'[2^P] > W'[2^P + 1] = 0.$$

Define $\delta_i = W'[i] - W'[i+1], i = 1, \ldots, 2^P$. We assume that:

$$\frac{\sum_{i=x,\ldots,2^P} \delta_i}{2^P - x + 1} \leq \frac{\sum_{i=1,\ldots,2^P} \delta_i}{2^P},$$

$2K < x < 2^P$ **(A2)**.

This assumption is based on the statistical results which indicate that δ_i tends to be relatively trivial when i is large enough (e.g., $i > 2K$). This can also be intuitively yet rationally illustrated as follows: Since $\delta i = W'[i] - W'[i+1] < W'[i]$, when i becomes larger, the value of $W'[i]$ gets smaller and, therefore, δ_i accordingly tends to be smaller. (A2) indicates that, since δ_i tends to get smaller as i grows larger, the mean value of the last $(2^P - x + 1)\delta_i$ (i.e., δ_i, $i = x, \ldots, 2^P$) tends to be smaller than the mean value of all δ_i (i.e., $i = 1, \ldots, 2^P$). Noting that $\sum_{i=N,\ldots,M} \delta_i \equiv W'[N] - W'[M+1]$ and $W'[2^P + 1] \equiv 0$, then we further have

$$(\mathbf{A2}) \Leftrightarrow \sum_{i=x,\ldots,2^P} \delta_i \leq \frac{2^P - x + 1}{2^P} \sum_{i=1,\ldots,2^P} \delta_i,$$

$$2K < x < 2^P \Leftrightarrow W'[x] \leq \frac{2^P - x + 1}{2^P} W'[1], 2K < x < 2^P.$$

Assumption 3. *Let β be the difference between the maximum and minimum numbers of key-ID groups allocated to the TCAM chips using the algorithm in this paper. In light of numerous experimental results, we find β is always small. Note that, in the typical case when $K = 5$ and $P = 4$, each TCAM gets only $2^P/K = 3.2$ key-ID groups on average and, as will be illustrated shortly, each TCAM at least gets two key-ID groups. Hence, we assume $\beta < 3$ in the following deduction* **(A3)**.

CFA Evaluation. As mentioned above, according to the FA scheme described in Section 3.4, the traffic load can be perfectly balanced when $Max_{k=1,\ldots,K} RM[k] \leq 2/K$, which is bound to be satisfied if the distributed table construction result is derived from CFA. Therefore, the throughput of DPPC-RE can be guaranteed to be K times that of the single chip solution.

LFA Evaluation. The performance evaluation for LFA is not as straightforward as that of CFA. Fortunately, we find that, by partitioning the entire process of LFA into two distinct phases, it is possible to deduce the worst-case performance phase by phase based on the three assumptions. First, recall (see Fig. 5) that, in each *round* of the LFA, K key-ID groups are allocated. Then, we denote by α the number of rounds where key-ID groups are allocated when the capacity constraint $Max_{k=1,\ldots,K} G[k] \leq C_t$ actually has no effect. For instance, according to A1, we have $\alpha \geq 2$, i.e., in the first two rounds of the LFA allocation, this constraint is always guaranteed to be satisfied.

Phase 1 (Rounds from 1 to α). *In this phase, the distribution process works without being restricted by the capacity constraint and the difference between the maximum and minimum $RM[i]$, denoted by Δ_I, accumulated to the end of this phase is no larger than $1/2^{P-1}$, i.e., $\Delta_I \leq 1/2^{P-1}$.*

Proof. Denote Δ_x as the difference between the maximum and minimum $RM[i]$ accumulated to the end of Round x, $1 \leq x \leq \alpha$. Note that, according to LFA, in Phase 1, each TCAM must be allocated exactly one key-ID group in each round and, for any pair of TCAMs, the one with a larger $RM[i]$ at the end of the previous round must be allocated a smaller $W'[i]$ in the current round. Thus, in any round x, $1 \leq x \leq \alpha$, the difference between allocated traffic load ratios of any two TCAMs must not accumulate from one round to the next and, therefore, Δ_x is no larger than either 1) the difference between the maximum and minimum $W'[i]$s within this round or 2) Δ_{x-1}, i.e., for $\forall x, 1 \leq x \leq \alpha$, $\Delta_x \leq Max_{i<x} W'[(i-1)K+1] - W'[iK]\}$ or $\Delta_x \leq \Delta_{x-1}$. Note that $\forall i$,

$$W'[(i-1)K+1] - W'[iK] < W'[(i-1)K+1] \leq 1/2^{P-1}$$

and $\Delta_1 = W'[1] - W'[K] \leq 1/2^{P-1}$; therefore, it can be justified, using mathematical induction, that $\forall x, 1 \leq x \leq \alpha \Delta_x \leq 1/2^{P-1}$. So, $\Delta_1 \equiv \Delta_\alpha \leq 1/2^{P-1}$. □

Phase 2 (The rest of the process). *In this phase, the capacity constraint works. Note, that until the end of Phase 1, each TCAM is allocated an identical number of key-ID groups. According to Assumption 3, the difference between the final numbers of allocated key-ID groups of any two TCAMs must be no more than 2. So, the worst case is that the first two, i.e., the largest two, $W'[\alpha K+1], W'[\alpha K+2]$ among the key-ID groups that have not been assigned yet are allocated to the TCAM chip with the largest traffic load ratio (accumulative to the end of Phase 1). Moreover, the final difference, denoted by Δ_2, may be even larger since the TCAM with the largest traffic load ratio may keep getting the relatively larger $W'[i]$ in the rest rounds. Note that this expansion of Δ_2 in each round must be no larger than $W'[\alpha K+3]$.So, in the worst case,*

$$\Delta_2 \leq \Delta_1 + W'[\alpha K+1] + W'[\alpha K+2] + \left\lfloor \frac{2^p - \alpha K - 2}{K} \right\rfloor \times W'[\alpha K+3].$$

Particularly, according to A1 and A2,

$$\begin{aligned} W'[\alpha K+1] + W'[\alpha K+2] &\leq \frac{2^P - (\alpha K+1) + 1}{2^P} W'[1] \\ &\quad + \frac{2^P - (\alpha K+2) + 1}{2^P} W'[1] \\ &\leq \frac{2^{P+1} - 2\alpha K - 1}{2^P} \times \frac{1}{2^{P-1}}. \end{aligned}$$

Similarly,

$$\left\lfloor \frac{2^P - \alpha K - 2}{K} \right\rfloor \times W'[\alpha K+3] \leq \left\lfloor \frac{2^P - \alpha K - 2}{K} \right\rfloor \times \frac{2^P - \alpha K - 2}{2^P} \times \frac{1}{2^{P-1}}.$$

So,

$$\Delta_2 \leq F(K,P,\alpha) = \frac{1}{2^{P-1}} \left(1 + \frac{2^{P+1} - 2\alpha K - 1}{2^P} + \left\lfloor \frac{2^P - \alpha K - 2}{K} \right\rfloor \times \frac{2^P - \alpha K - 2}{2^P} \right). \quad (1)$$

Note that the objective $Max_{j=1,\ldots,K} RM[j]$ is no larger than that when: 1) The traffic load ratios of the rest of the $K-1$ TCAMs are all Δ_2 less than the maximum one and 2) $\alpha = 2$, since $\frac{\partial F(K,P,\alpha)}{\partial \alpha} < 0$ and $\alpha \leq 2$. In this case, the following equality holds:

$$Max_{i=1,\ldots,K} RM[i] + (K-1)(Max_{i=1,\ldots,K} RM[i] - \Delta_2) = 1.$$

Therefore, we get, in the worst case,

$$Max_{i=1,\ldots,K} RM[i] = \frac{1}{K} + \frac{K-1}{K} \Delta_2. \quad (2)$$

According to (1), for the typical case when $P=4$ and $K=5$, we have $\Delta_2 \leq 21.88\%$ and, according to (1), $Max_{i=1,\ldots,K} RM[i] \leq 37.50\% \leq 40\% = 2/K$. It means that, based on the FA scheme described in Section 3.4, the traffic load can also be perfectly balanced if the distributed table construction is derived from LFA, under the three reasonable assumptions. Therefore, the throughput of DPPC-RE can be guaranteed to be K times that of the single chip solution (K ppc).

5.2 Worst-Case Processing Delay

We derive a performance upper bound for DPPC-RE. Note that the maximum delay of the process is a very important measure of the DPPC-RE scheme, which impacts overall fast data path performance in terms of packet delay and loss (note that DPPC-RE may cause a slight loss of packets when the system is heavily loaded). Fortunately, we find that the processing delay of the DPPC-RE scheme is upper bounded. Denote the depth of the RM FIFO and the KE FIFO as D_r and D_k, respectively, the round-robin-ratio (defined in Section 4) as n, the TCAM cycle as T_c, and the maximum processing latency of a key-encoding operation as T_k. Then, we have:

Theorem 2. $T_k \leq D_k \times 2(n+1) \times T_c$.

Proof. According to the weighted round-robin mechanism (described in Section 4.3), the key-encoding tasks, if there are any, will be processed in at most every $n+1$ turns (each turn is either a rule-matching or key-encoding operation). Since each rule-matching or key-encoding operation takes two TCAM access cycles, the head unit of a KE FIFO would be processed within $2(n+1)T_c$. Note that the depth of the KE FIFO is D_k. Hence, in the worst case, the tail unit in the KE FIFO needs to wait $D_k \times 2(n+1)$ TCAM cycles until the corresponding key-encoding task finishes. Therefore, $T_k \leq D_k \times 2(n+1) \times T_c$. □

Theorem 3. *Suppose that, at a specific instant T_n, the key-encoding results for the first q units in an RM FIFO are all returned. Let the time instant when the rule-matching task of the qth unit finishes be T_{finish}. Then, $T_{finish} \leq T_n + (q + \lfloor q/n \rfloor + 1) \times 2 \times T_c$.*

Proof. Since the key-encoding results for the first q units are all returned, there is no need for rule-matching to wait for key-encoding results until time T_{finish}. According to the weighted round-robin mechanism, in the worst case, q rule-matching tasks would be interrupted by at most $\lfloor q/n \rfloor + 1$ key-encoding tasks. Note that each rule-matching or key-rncoding operation takes two TCAM access cycles. So, in this case, $(q + \lfloor q/n \rfloor + 1) \times 2 \times T_c$ is needed until the rule-matching task of the qth unit finishes. Hence, $T_{finish} \leq T_n + (q + \lfloor q/n \rfloor + 1) \times 2 \times T_c$.□

Theorem 4. *Let T_d denote the processing delay of an operation. Then, T_d satisfies:*

$$T_d \leq [D_k \times 2 \times (n+1) + (D_r + \lfloor D_r/n \rfloor + 1) \times 2] \times T_c.$$

Proof. First, the processing delay of an operation is actually the processing delay of the corresponding rule-matching operation (the last step of the operation). For any given packet $P0$, suppose that the instant it arrives is T_a and, at that instant, there are $q-1$ units in the RM FIFO at the TCAM where the rule-matching task for $P0$ is to be performed (namely, it is the qth unit). Note that the instant when all the key-encoding results for the first q units in this RM FIFO are returned (denoted as T_n) is actually the instant when the last key-encoding result for the first q units is returned (denoted as T_n'). According to Theorem 2, $T_n' - T_a \leq D_k \times 2(n+1) \times T_c$, so $T_n = T_n' \leq T_a + D_k \times 2(n+1) \times T_c$. Suppose that, at the instant T_n, packet $P0$ is in the q'th position. According to Theorem 3, we further have

$$T_{finish} \leq T_n + (q' + \lfloor q'/n \rfloor + 1) \times 2 \times T_c \leq T_a + D_k \times 2 \times (n+1) \times T_c + (q' + \lfloor q'/n \rfloor + 1) \times 2 \times T_c.$$

Note that $q' \leq D_r$; therefore,

$$T_{finish} \leq T_a + [D_k \times 2(n+1) + (D_r + \lfloor D_r/n \rfloor + 1) \times 2] \times T_c.$$

Therefore,

$$T_d = T_{finish} - T_a \leq [D_k \times 2 \times (n+1) + (D_r + \lfloor D_r/n \rfloor + 1) \times 2] \times T_c.$$

□

For the typical case when $D_k = 4$, $D_r = 8$, $n = 3$, we have $T_d \leq 54 \times T_c$.

5.3 Power Efficiency

TCAM is a fully associated memory. When the search key is presented at the input, all TCAM entries are triggered simultaneously for the matching. This, therefore, results in very high power consumption. Hence, power consumption for TCAM has traditionally been one of the major concerns. With the improvement of TCAM technique, some of the latest products, e.g., the Cypress Ayama 10K/20K NSE Series TCAM [20], have the ability (powered by the function called *MiniKey*) to let the designer categorize the entries into groups. For each matching, by providing the ID of the entry group, the matching can be limited within the corresponding entry group, i.e., only the entries belonging to this group will be triggered, hence saving power. Many works have been proposed based on such a technique [22], [26], [27]. For instance, Zane et al. [22] developed a power efficient route lookup engine, called CoolCAM. Benefiting from such new features of TCAMs, CoolCAM uses both trie-based and bit-selection-based table partitioning schemes to reduce the number of entries triggered during a TCAM match and achieves a power reduction factor of about 7.55 when adopting eight TCAM partitions.

Compared with CoolCAM, which relies only on the new features of TCAM, DPPC-RE comes with the ability to further optimize TCAM power efficiency and achieves a better power reduction factor. First, according to the DTCS scheme, each TCAM contains only a small portion of the policy table, including several Key-ID groups. Hence, for each matching operation, merely a portion of the policy rules would be matched. Supposing that the rules are evenly allocated among the TCAM chips, the power consumption is reduced by a factor of $\frac{N}{D(N_e,K)/K} = \frac{N \times K}{DER \times N_e}$. Second, with the *MiniKey* function, we may simply let the rules in each key-ID groups be an entry group. Then, for each matching, the power consumption can be reduced by a factor of $\frac{N}{D(N_e,K)/2^P} = \frac{N \times 2^P}{DER \times N_e}$, e.g., = 12.5 in the example described in Section 3, where $P = 4$, $N = 2{,}180$, $N_e = 1{,}550$, $K = 5$, and $DER = 1.80$, assuming that the sizes of different Key-ID groups are similar.

5.4 Database Updating

A rule table is dynamically managed by adding new rules and/or deleting old rules from time to time. Moreover, these rule updates may result in an encoded range table update to maximize the TCAM storage efficiency. Hence, how to effectively handle rule and range table update is a key issue in the proposed algorithms. In [19], we proposed a lock-free algorithm, called the Consistent Policy Table Update Algorithm (CoPTUA), for rule update in a single TCAM. The basic idea of CoPTUA is to maintain a consistent and error-free rule table during the rule update process, thus eliminating the need for TCAM rule table locking while ensuring the correct rule-matching. CoPTUA is further extended to design a lock-free algorithm for range table update in [15]. In this section, we show that the lock-free rule and range table update algorithms can be easily incorporated into DPPC-RE, thus eliminating the possible negative performance impact of range and rule table update. In DPPC-RE, a rule may be placed in 1) only one TCAM or 2) multiple TCAMs, as we stated in Section 2. In case 1, the rule update (addition or deletion) process is the same as that in the single TCAM case [19]. In case 2, the rule update may not be finished at the same time in different TCAMs. However, any search key can only be classified in one TCAM, which is maintained as a consistent table during the update process, and, hence, no inconsistency is introduced due to the different activating time of the updated rule in different TCAMs. The encoded range update process in DPPC-RE is similar to that of the single TCAM case described in [15]. The only difference is that the range tables are duplicated in every TCAM. What we need to keep in mind is that any rule table can be updated only after all range tables are updated for encoding a newly selected range and any range table can be updated only

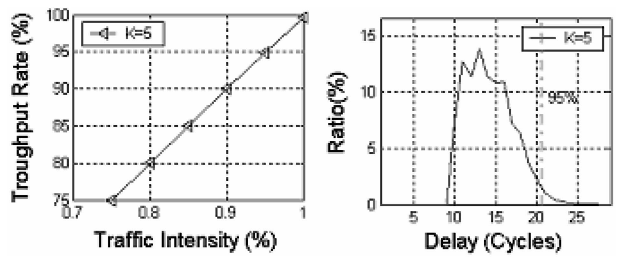


Fig. 9. Simulation results (throughput and delay).

after all rule tables are updated by unencoding an encoded range. It is not difficult to realize that the consistency of rule and range tables is maintained during the encoded range update process.

The rule update may result in a variation in lookup throughput and the number of rules among different TCAMs. We set two thresholds for each TCAM. One is lookup throughput, which is determined based on the worst-case analysis. When the lookup throughput in a TCAM reaches the threshold value, a rule group with heavy lookup throughput will be exchanged with a rule group with relatively light lookup throughput from another TCAM. If no such rule group can be exchanged due to space capacity constraint, the LFA-CFA algorithm will be run to reconstruct the tables. The other threshold is the number of rules in a TCAM. Similarly to the single TCAM case, some empty rule entries have to be kept for rule update purpose. When the number of empty rule entries is less than the threshold, e.g., 1 percent C_t, a rule group of a large number of rules will be exchanged with a rule group of a small number of rules from the TCAM with the least number of rules, while maintaining the lookup threshold of each TCAM within a given value. If no rule group can be exchanged, the LFA-CFA will be run to reconstruct the table. Due to the slow rule update rate (from once in a few seconds to once in a few days), the reconstruction rate is low.

5.5 Simulation Results

The performance of the proposed solution is evaluated by simulation. We assume the traffic arrival process is Poisson and the traffic load distribution among key-ID groups is given in Fig. 6. The following parameters are used in the simulation: $K = 5$, RM FIFO queue size = 8, Key Buffer size = 8, KE FIFO queue size = 4, and Round-Robin-Ratio = 3.

5.5.1 Throughput and Delay Performance

The simulation results are given in Fig. 9. First of all, we find that the throughputs are perfectly guaranteed even when the system is heavily loaded (traffic intensity tends to 100 percent, i.e., 125 Mpps. In the case when $K = 5$ and 100 MHz TCAM chips are adopted, this index actually equals 5 ppc), showing that the traffic load is well balanced by the DPPC-RE mechanism. The minimum delay for each packet classification is 10 TCAM cycles (five for rule-matching and five for key-encoding). In general, however, additional cycles are needed because of the queuing effect. We focus on the performance when the system is heavily loaded. The lower two subplots in Fig. 9 show the delay distribution for the back-to-back mode (i.e., traffic intensity = 100 percent). We find that more than 95 percent of the packets are processed within 21 (TCAM) cycles. Suppose that the TCAM chips work at 100MHz and the delay is around 210~260 ns for the majority of the packets, indicating that both delay and delay jitters are reasonably small. Also, note that we have proven in Section 5.2 that the worst-case delay for each packet classification has an upper bound of about 50 TCAM cycles, i.e., 500 ns.

TABLE 4
Simulation Results of Changing Traffic Pattern

Case	Table constructed from	Traffic change to	Before (%)	After (%)
I	Pattern I	Pattern II	99.76	99.96
II	Pattern II	Pattern I	100	99.63

5.5.2 Impact of Traffic Pattern Changes

In order to measure the stability and adaptability of the DPPC-RE scheme when the traffic pattern changes over time, we run the following simulations at the back-to-back mode (i.e., traffic intensity = 100 percent). The traffic pattern depicted in Fig. 6 is denoted as Pattern I (uneven distribution) and the uniform distribution is denoted as Pattern II. We first construct the distributed table according to one of the patterns and measure the throughput performance under this traffic pattern. Then, we change the traffic to the other pattern and get the throughput performance again without reconstructing the distributed table. The associated simulation setups and the results are given in Table 4. We find that, although the traffic pattern changes significantly, the throughput performance just decreases slightly[6] (<1 percent) in all the cases. This means that FA excels in adaptive load balancing and the DPPC-RE scheme copes with the changes of traffic pattern well.

5.6 Comparison with Other Schemes

In this part, we assume that common 100 MHz TCAM chips are adopted to implement the packet classification engine since a single 100MHz TCAM chip cannot provide OC768 wire speed. So, CLP must be adopted to achieve this goal. Depending on the method of achieving CLP (to use distributed storage or to duplicate the table) and, whether we adopt key-encoding or not, there would be four different possible schemes. They are:

1. *Duplicate Table + No Key-Encoding*. Two 100 MHz TCAM chips should be used in parallel to achieve 125 Mpps, with each containing a full, unencoded rule table (with N entries). A total number of $K \times N$ TCAM entries are required. It is the simplest one to implement and offers deterministic performance (i.e., with zero loss rate and fixed processing delay).
2. *Duplicate Table + Key-Encoding*. Five 100 MHz TCAM chips should be used in parallel to achieve 125 Mpps, with each containing an encoded rule table (with N_e entries). A total number of $K \times N_e$ TCAM entries

6. In Case 1, the throughput even increases, which indicates that the change of the pattern even has a positive effect on the performance. This may be caused by the use of the greedy (i.e., not optimum) algorithm for table construction.

TABLE 5
Comparison of Expansion Ratios

Database	#1	#2	#3	#4	#5
Original Rules (N_0)	279	183	158	264	1550
Expanded Rules (N)	949	553	415	1638	2180
After KE (N_e)	279	183	158	264	1550
Expansion Ratio to Support OC768					
Duplicate+NoKE(K=2)	6.80	6.04	5.98	12.40	2.82
Duplicate+KE(K=5)	5.08	5.11	5.10	5.14	5.03
Distributed+KE (K=5)	1.59	1.70	1.82	2.19	1.53
Distributed+NoKE(K=2)	5.18	4.46	3.43	7.81	1.69

are required. It also offers lossless, deterministic performance.

3. *Distributed Storage + Key-Encoding (DPPC-RE)*. Five 100 MHz TCAM chips should be used in parallel. The total number of TCAM entries required is $N_e \times DER$ (which is not linearly proportional to K). It may incur slight loss when heavily loaded.
4. *Distributed Storage + No Key-Encoding*. Two 100 MHz TCAM chips should be used in parallel. The total number of TCAM entries required is $N \times DER$. Without a dynamic load balancing mechanism (which can only be employed when adopting key-encoding), its performance is undeterministic and massive loss may occur when the system is heavily loaded or the traffic pattern changes.

The TCAM Expansion Ratio *ERs* (defined as the ratio of the total number of TCAM entries required to the total number of rules in the rule database) are calculated for all the five real-world databases based on these four schemes. The results are given in Table 5. Apparently, Distributed + KE, or DPPC-RE, significantly outperforms all three of the other schemes in terms of the TCAM storage efficiency.

Obviously, DPPC-RE exploits the trade-off between deterministic performance and high statistical throughput performance, while the schemes with table duplication gain high deterministic performance at significant memory cost. The combination of distributed storage and range encoding (i.e., the DPPC-RE scheme) provides a very good balance in terms of the worst-case performance guarantee and low memory cost, making it an attractive solution.

On the other hand, we should also notice that, although DPPC-RE provides a good trade-off between storage requirement and scalable performance, it requires additional specific hardware assistance. Moreover, the mapper should work at the speed K times faster than that of the memory chips (e.g., as far as 100 MHz TCAM is concerned, the mapper should work at approximately K^*100 MHz). Fortunately, thanks to modern ASIC technology, which can work at multiple GHz clock rate [21], the mapper, which is simple in function, can be easily implemented with low cost.

6 Conclusion

Insufficient memory bandwidth for a single TCAM chip and large expansion ratio caused by the range matching problem are the two important issues that have to be solved when adopting TCAM to build a high performance and low cost packet classifier for next generation multigigabit router interfaces. In this paper, a distributed parallel packet classification scheme with range encoding (DPPC-RE) is proposed to achieve OC768 wire speed packet classification with minimum TCAM cost. DPPC-RE includes a rule partition algorithm to distribute rules in different TCAM chips with minimum redundancy and a heuristic algorithm to balance the traffic load and storage demand among all TCAMs. The implementation details are given. A thorough theoretical worst-case analysis of throughput, processing delay, and power consumption, as well as the experimental results, shows that the proposed solution can provide scalable throughput (e.g., 125 Mpps in the presented prototype), matching OC768 line rate or even higher, with, e.g., 50~119 percent additional TCAM resource, found for the five real-world classifiers.

Acknowledgments

The authors would like to express their gratitude to Professor Jonathan Turner and Dr. David Taylor from Washington University in St. Louis for kindly sharing their real-world databases and the related statistics with them. The authors thank the two reviewers for their constructive comments that significantly strengthen the paper quality. Finally, the authors also thank the reviewers of IEEE INFOCOM 2005 for their valuable comments on the short version of this paper. This work was supported by the NSFC (No.60373007 and No. 60573121), China/Ireland Science and Technology Collaboration Research Fund (CI-2003-02), Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048), and the US National Science Foundation under grants ANI-0125653 and ANI-9809533. Part of this work was published in the *Proceedings of IEEE INFOCOM 2005* [24].

References

[1] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and Scalable Layer Four Switching," *Proc. ACM SIGCOMM Conf.*, 1998.
[2] T.V. Lakshman and D. Stiliadis, "High-Speed Policy-Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," *Proc. ACM SIGCOMM Conf.*, 1998.
[3] M. Buddhikot, S. Suri, and M. Waldvogel, "Space Decomposition Techniques for Fast Layer-4 Switching," *Proc. Conf. Protocols for High Speed Networks IV (PfHSN '99)*, 1999.
[4] A. Feldmann and S. Muthukrishnan, "Tradeoffs for Packet Classification," *Proc. IEEE INFOCOM Conf.*, 2000.

[5] F. Baboescu, S. Singh, G. Varghese, "Packet Classification for Core Routers: Is There an Alternative to CAMs?" *Proc. IEEE INFOCOM Conf.,* 2003.

[6] P. Gupta and N. McKeown, "Packet Classification on Multiple Fields," *Proc. ACM SIGCOMM Conf.,* 1999.

[7] P. Gupta and N. McKeown, "Packet Classification Using Hierarchical Intelligent Cuttings," *IEEE Micro Magazine,* vol. 20, no. 1, pp. 34-41, Jan.-Feb. 2000.

[8] V. Srinivasan, S. Suri, and G. Varghese, "Packet Classification Using Tuple Space Search," *Proc. ACM SIGCOMM Conf.,* 1999.

[9] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet Classification Using Multidimensional Cutting," *Proc. ACM SIGCOMM Conf.,* 2003.

[10] E. Spitznagel, D. Taylor, and J. Turner, "Packet Classification Using Extended TCAMs," *Proc. 11th Int'l Conf. Network Protocol (ICNP '03),* 2003.

[11] T. Lakshman and D. Stiliadis, "High-Speed Policy-Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," *ACM SIGCOMM Computer Comm. Rev.,* vol. 28, no. 4, pp. 203-214, Oct. 1998.

[12] H. Liu, "Effcient Mapping of Range Classier into Ternary CAM," *Proc. 10th Symp. High Performance Interconnects (HotI '02),* Aug. 2002.

[13] J. van Lunteren and A.P.J. Engbersen, "Dynamic Multi-Field Packet Classification," *Proc. Globecom '02 Conf.,* pp. 2215-2219, Nov. 2002.

[14] J. van Lunteren and A.P.J. Engbersen, "Fast and Scalable Packet Classification," *IEEE J. Selected Areas in Comm.,* vol. 21, no. 4, pp. 560-571, May 2003.

[15] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES: Dynamic Range Encoding Scheme for TCAM Coprocessors," technique report, Univ. of Texas at Arlington, http://crystal.uta.edu/hche/dres.pdf, 2006.

[16] K. Zheng, C.C. Hu, H.B. Lu, and B. Liu, "An Ultra High Throughput and Power Efficient TCAM-Based IP Lookup Engine," *Proc. IEEE INFOCOM Conf.,* Apr. 2004.

[17] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach,* third ed. Morgan Kaufmann, 2002.

[18] J. Turner, "Real-World Rule Databases," St. Louis, Mo.: Washington Univ., 2004.

[19] Z. Wang, H. Che, M. Kumar, and S.K. Das, "CoPTUA: Consistent Policy Table Update Algorithm for TCAM without Locking," *IEEE Trans. Computers,* vol. 53, no. 12, pp. 1602-1614, 2004.

[20] Cypress Ayama 10K/20K NSE Series TCAM products, http://www.cypress.com, 2006.

[21] Intel Pentium IV Series CPU products, http://www.intel.com, 2006.

[22] F. Zane, G. Narlikar, A. Basu, "CoolCAMs: Power-Efficient TCAMs for Forwarding Engines," *Proc. IEEE INFOCOM '03,* 2003.

[23] Cisco CRS-1 Carrier Routing System, http://www.cisco.com, 2006.

[24] K. Zheng, H. Che, Z. Wang, and B. Liu, "TCAM-Based Distributed Parallel Packet Classification Algorithm with Range-Matching Solution," *Proc. IEEE INFOCOM '05,* vol. 1, pp. 293-303, Mar. 2005.

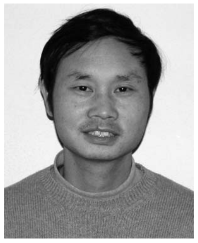
[25] H. Lu and S. Sahni, "O(log W) Multidimensional Packet Classification," *IEEE/ACM Trans. Networking,* to appear.

[26] H. Lu, "Improved Trie Partitioning for Cooler TCAMs," *Proc. IASTED Int'l Conf. Advances in Computer Science and Technology (ACST),* 2004.

[27] R. Panigrahy and S. Sharma, "Reducing TCAM Power Consumption and Increasing Throughput," *Proc. 10th Symp. High Performance Interconnects HOT Interconnects (HotI '02),* 2002.

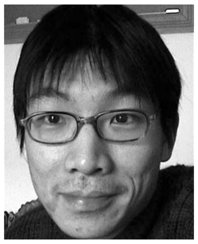
Kai Zheng received the BS degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2001. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Tsinghua University. His research interests include IP address lookup, packet classification, and pattern matching associated network security issues. He is a student member of the IEEE.

Hao Che received the BS degree from Nanjing University, Nanjing, China, in 1984, the MS degree in physics from the University of Texas at Arlington in 1994, and the PhD degree in electrical engineering from the University of Texas at Austin in 1998. He was an assistant professor of electrical engineering at Pennsylvania State University, University Park, from 1998 to 2000, and a system architect with Santera Systems, Inc., Plano, Texas, from 2000 to 2002. Since September 2002, he has been an assistant professor of computer science and engineering at the University of Texas at Arlington. His current research interests include network architecture and design, network resource management, multiservice switching architecture, and network processor and coprocessor design. He is a member of the IEEE.



Zhijun Wang received the PhD degree in computer science and engineering from the University of Texas at Arlington. Currently, he is an assistant professor in the Department of Computing, the Hong Kong Polytechnic University, Hong Kong. His current research interests include high-speed networks, network security, data management in mobile networks, and peer-to-peer networks.

Bin Liu received the MS and PhD degrees, both in computer science and engineering, from Northwestern Polytechnical University, Xi'an, China in 1988 and 1993, respectively. From 1993 to 1995, he was a postdoctoral research fellow in the National Key Lab of SPC and Switching Technologies, Beijing University of Post and Telecommunications. In 1995, he transferred to the Department of Computer Science and Technology, Tsinghua University as an associate professor, where he mainly focused on multimedia networking including ATM switching technology and Internet infrastructure. He became a full professor in the Department of Computer Science and Technology at Tsinghua University, Beijing in 1999 and currently he is the director of the Lab of Broadband Networking Technologies at the university. His current research areas include high-performance switches/routers, high-speed network security, network processors, and traffic management. He is a member of the IEEE.



Xin Zhang is a fourth-year undergraduate in the Department of Automation, Tsinghua University. He is expected to receive the BS degree by July 2006 and to pursue the PhD degree in the United States. His current researches focus on IP route lookup, packet classification, as well as system and algorithm design.