TCAM-based Distributed Parallel Packet Classification Algorithm with Range-Matching Solution[i]

Kai Zheng[1], Hao Che[2], Zhijun Wang[2], Bin Liu[1]
[1]The Department of Computer Science, Tsinghua University, Beijing, P.R.China 100084
[2]The Department of Computer Science and Engineering; The University of Texas at Arlington, Arlington, TX 76019, USA
zk01@mails.Tsinghua.edu.cn, {hche, zwang}@cse.uta.edu, liub@Tsinghua.edu.cn

***Abstract*--Packet Classification (PC) has been a critical data path function for many emerging networking applications. An interesting approach is the use of TCAM to achieve deterministic, high speed PC. However, apart from high cost and power consumption, due to slow growing clock rate for memory technology in general, PC based on the traditional single TCAM 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 PC throughput. 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. Using commercially available TCAM chips, the proposed scheme achieves PC performance of more than 100 million packets per second (Mpps) , matching OC768 (40 Gbps) line rate.**

***Key words*—System Design, Simulations**

I. INTRODUCTION

Packet Classification (PC) 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. PC involves various matching conditions, e.g., longest prefix matching (LPM), exact matching, and range matching, making it a complicated pattern matching issue. Moreover, since PC lies in the critical data path of a router and it has to act upon each and every packet at wire-speed, this creates a potential bottleneck in the router data path, particularly for high speed interfaces. For example, at OC192 (10 Gbps) full line rate, a line card (LC) needs to process about 25 million packets per second (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 an LC is moving towards OC768, it poses significant challenges for the design of packet classifiers to allow wire-speed forwarding.

The existing algorithmic approach including geometric algorithms based on the hierarchical trie [1] [2] [3] [4] and most heuristic algorithms [6] [7] [8] [9] generally require nondeterministic number of memory accesses for each lookup, which makes it difficult to use pipeline to hide the memory access latency, limiting the throughput performance. Moreover, most algorithmic approaches, e.g., geometric algorithms, apply only to 2-dimensional cases. Although some heuristic algorithms address higher dimensional cases, they offer nondeterministic performance, which differs from one case to another.

In contrast, ternary content addressable memory (TCAM) based solutions are more viable to match high speed line rates, while making software design fairly simple. A TCAM finds a matched rule in 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 OC192 to OC768 line rates. For example, for a TCAM with 100 MHz clock rate, it can perform 100 million (M) TCAM lookups per second. Since each typical 5-tuple policy table matching requires two TCAM lookups, as will be explained in detail later, the TCAM throughput for the 5-tuple matching is 50Mpps. As aforementioned, to keep up with OC192 line rate, PC has to keep up with 25Mpps lookup rate, which translates into a budget of two 5-tuple matches per packet. The budget reduces to 0.5 matches per packet at OC768. Apparently, with LPM and firewall/ACL competing for the same TCAM resources, it would be insufficient using a single 100 MHz TCAM for PC while maintaining OC192 to OC768 line rates. Although increasing the TCAM clock rate can improve the performance, it is unlikely that a TCAM technology that matches the OC768 line speed will be available anytime soon, given that the memory speed improves by only 7% each year [17].

Instead of striving to reduce the access latency for a single TCAM, a more effective approach is to exploit chip-level parallelism (CLP) to improve overall PC throughput performance. However, a naive approach to realize CLP by simply duplicating the databases to a set of uncoordinated TCAM chips can be costly, given that TCAM is an expensive commodity. In a previous work [16] by two of the authors of the present work, 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 for PC is its ability to effectively solve the range matching problem. [10] reported that today's real-world policy filtering (PF) tables involve significant percentages of rules with ranges. Supporting rules with ranges or range matching in TCAM can lead to very low TCAM storage efficiency, e.g., 16% as reported in [10]. [10] proposed an extended TCAM scheme to improve the TCAM

[i]This research is supported by the NSFC (No.60173009 & No.60373007) and the National 863 High-tech Plan (No.2003AA115110 & No. 2002AA103011-1).

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 were also reported [11] [12] [13] [14] [15]. A key challenge for range encoding is the need to encode multiple subfields in a search key extracted from the packet to be classified at wire-speed. To achieve high speed search key encoding, parallel search key sub-field encoding were 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 to use 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 PC 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; 2) it involves three different matching conditions including prefix, range, and exact matches.

In this paper, we propose an efficient TCAM CLP scheme, called Distributed Parallel PC with Range Encoding (DPPC-RE), for the typical 5-tuple PC. 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], both a static algorithm and a fully adaptive algorithm are proposed to deal with range encoding and load balancing simultaneously. The simulation results show that the proposed solution can achieve 100 Mpps throughput performance matching OC768 line rate, with just 50% additional TCAM resource compared with a single TCAM solution at about 25 Mpps throughput performance.

The rest of the paper is organized as follows. Section II gives the definitions and theorems which will be used throughout the paper. Section III presents the ideas and algorithms of the DPPC-RE scheme. Section IV presents the implementation details on how to realize DPPC-RE. The performance evaluation of the proposed solution is given in Section V. Finally, Section VI concludes the paper.

II. DEFINITIONS AND THEOREMS

Rules: A rule table or policy filtering table includes a set of *match conditions* and their corresponding *actions*. We consider the typical 104-bit five-tuple match conditions, i.e., (SIP(1-32), DIP(1-32), SPORT(1-16), DPORT (1-16)，PROT(1-8)[ii]), 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 longest prefix matching (LPM); SPORT and DPORT generally require range matching; and PROT requires exact matching. Except for sub-fields with range matching, any other sub-field in a match condition can be expressed using a single string of ternary bits, i.e., 0, 1, or "don't care" *. Table I gives an example of a typical five-tuple rule table.

TABLE I An Example of Rule Table with 5-tuple Rules

	Src IP	Dst IP	Src Port	Dst Port	Prot		Action
L1	1.1.*.*	2.*.*.*	*	*	6	→	AF
L2	2.2.2.2	1.1.*.*	*	256-512	6	→	BF
L3	3.3.*.*	*.*.*.*	>1023	512-1024	11	→	EF
L4	*.*.*.*	4.4.4.*	5000-6000	>1023	*	→	Accepted
L5	*.*.*.*	*.*.*.*	<1023	*	*	→	Discard
…	……	……	……	……	……	→	……

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

Rule Entry: TCAMs are organized in slots with fixed size (e.g., 64 or 72); each rule entry takes 1 or more slots depending on its size. Fig. 1 shows the implementation of rule L_1 and L_2 in TCAM with 64-bit slots. Rule L_1 has no range in any of its subfields and hence it takes 2 slots with 24 free bits in the second slot. Each of such rules 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 sub-field. This range cannot be directly expressed as a string of ternary bits, and must be partitioned into two sub-ranges: {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 strings is defined as the *non-Trivial Range*. Hence, *L*2 takes 4 slots (slots 3, 4, 5 and 6), or 2 rule entries in the TCAM.

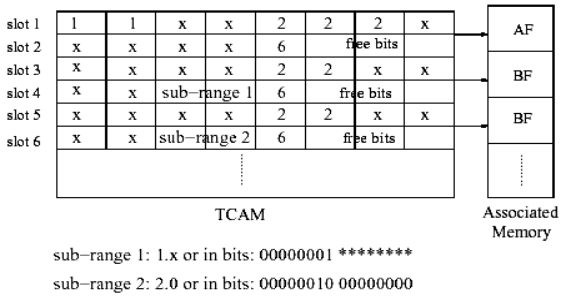

Fig. 1 Rules in a TCAM. The range {256-512}is split into 2 sub-ranges {256-511}and {512}, and implemented as sub-range 1 and sub-range 2. '*' represents a 'don't care' bit, and ' x '=' ******** ', a wildcard byte. The other numbers represent the actual byte values.

In general, if ranges in the SPORT and DPORT sub-fields in a match condition take *n* and *m* ternary strings, respectively, the match condition takes up $n \times m$ 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. After range encoding, a rule with

[ii] The bits in the sub-field are ordered with the 1st bit (MSB) lies in the leftmost position.

encoded ranges only takes one rule entry, thus significantly improving TCAM storage efficiency.

Let N_0 denote the rule table size, or the number of rules in a rule table; N represent the number of TCAM entries required to accommodate the rule table without range encoding; N_e stand for the number of TCAM entries required to accommodate the rule table with range encoding.

Search Key: A search key is a 104 binary bit string composed of a five-tuple. For example, <1.1.1.1, 2.2.2.2, 1028, 34556, 11> is a five-tuple search key. In general, a search key is extracted by a network processor from the IP header and passed to a packet classifier to match against a five-tuple rule table.

Matching: In the context of TCAM based PC 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 search key and the match condition, either of the following two conditions is met: (1) the bit values are identical; (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 PC is developed.

ID: The idea of the proposed distributed TCAM PC is to make use of a small number of bit values extracted from certain bit positions in the search key and match condition as IDs to (1) divide match conditions or rules into groups, which are mapped to different TCAMs; (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 sub-fields of a match condition as the rule ID, denoted as Rule-ID, for the match condition and use *P* number of bits extracted from the corresponding search key positions as the key ID, denoted as 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.*.*, 2.*.*.*, *, *, 6> is"01*0"and the key-ID for the search key <1.1.1.1, 2.2.2.2, 1028, 34556, 11> 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

$$KG_i = \bigcup_{RID_j \text{ match } KID_i} RG_j \quad .$$

For example, for *P=3,* the Key-ID group "011" is composed of the following 8 Rule-ID groups: "011，*11，0*1，01*，**1，*1*，0**，***".

An immediate observation is that different key-ID groups may overlap with one another in the sense that different key-ID groups may have common Rule-ID groups.

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 rules 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. *Throughput* is defined as the number of PCs per unit time. It is an important measure of the processing power of the proposed solution. *Traffic intensity* is used to characterize the workload in the system. As the design is targeted at PC 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., 100 Mpps. *Throughput ratio* is defined as the ratio between *Throughput* and the worst-case traffic load at OC768 line rate.

Now, two theorems are established, which state under what conditions the proposed solution ensures correct rule matching and maintains the original ordering of the packets, respectively.

Theorem 1: For each PC, correct rule matching is guaranteed if

a) All the rules belonging to the same Key-ID group are placed in the same TCAM with correct priority orders.

b) A search key containing a given Key-ID is matched against the rules in the TCAM, in which the corresponding Key-ID group is placed.

Proof: On the one hand, a necessary condition for a given search key to match a rule is that the Rule-ID for this rule matches the Key-ID for the search key. On the other hand, any rule that does not belong to this Key-ID group cannot match the search key, because the Key-ID group contains all the rules that match the Key-ID. Hence, a rule match can occur only between the search key and the rules belonging to the Key-ID group corresponding to the search key. As a result, meeting conditions a) and b) will guarantee the correct rule matching □

Theorem 2: 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 five-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. □

III. ALGORITHMS AND SOLUTIONS

The key problems we aim to solve are 1) how to make use of CLP to achieve high performance with minimum cost; 2) how to solve the TCAM range matching issue 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 PC throughput performance without incurring much additional cost.

The idea of RE is to encode the range sub-fields of the rules and the corresponding sub-fields in a search key into bit-vectors, respectively. In this way, the number of ternary strings (or TCAM entries, which will be defined shortly in Section III.C) required to express a rule with non-trivial 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, making DPPC-RE indeed a practical solution. In what follows, we introduce DPPC-RE in detail.

A. ID Bits 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 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 5 real-world database analyses [18] that are used throughout the rest of the paper to simplify the computation as follows:

1) Since the sub-fields, DPORT and SPORT, in a rule may have non-trivial ranges which need to be encoded, we choose not to take these two sub-fields into account for ID-bit selection;

2) According to the analysis of several real-world rule databases [18], over 70% rules are with non-wildcarded PROT sub-field, and over 95% of these non-wildcarded PROT sub-fields are either TCP(6) or UDP(11) (approximately 50% are TCP). Hence, one may select either the 8th or the 5th bit (TCP and UDP PROTs have different values at these two bit positions) of the PROT sub-field as one of the ID bits. All the rest of the bits in the PROT sub-field have fixed one-to-one mapping relationship with the 8th or 5th bits, and do not lead to any new information about the PROT;

3）Note that the rules with wildcard(s) in their Rule-IDs are actually those incurring redundant storage. The more the 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 5 real-world rule databases, there are over 92% rules whose DIP sub-fields are prefixes no longer than 25 bits and there are over 90% rules whose SIP sub-fields 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 sub-fields, since they are wildcards in most cases.

Based on these 3 empirical rules, the traversal is simplified as: choose an optimal (P-1)-bit combination out of 50 bits of DIP and SIP sub-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] (with 1550 total number of rules). 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) [iii].

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.

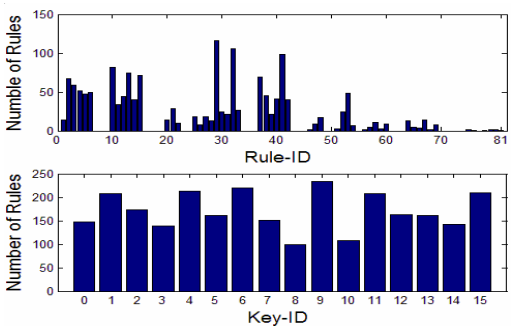

Fig. 2 ID-bit Selection Result of Rule Database Set #5.

B. 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 the TCAM chips. For clarity, we first describe the mathematical model of the distributed table construction problem as follows.

Let:

Q_k be the set of the Key-ID groups placed in TCAM #k where $k=1,2, \ldots, K$;

$W[j], j=1,\ldots,2^P$ 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

[iii] The leftmost is the least significant

TCAM #k, namely, $RM[k] := \sum_{j \in Q_k} W[j]$;

$G[k]$ be the number of rules distributed to TCAM #k, namely, $G[k] := |\bigcup_{KG_i \in Q_K} KG_i|$.

C_t be the capacity tolerance of the TCAM chips (the maximum number of rules it can contain), and L_t be the tolerance (maximum value) of the traffic load ratio that a TCAM chip is expected to bear.

The optimization problem for distributed table construction is given by:

==========

Find a *K*-division $\{Q_k, k = 1,...,K\}$ of the Key-ID groups that

Minimize:

$$\underset{k=1,...,K}{Max} RM[k]; \quad \underset{k=1,...,K}{Max} G[k];$$

Subject To:

$$Q_k \in S, \ \bigcup_{k=1,...,K} Q_k = S;$$

$$RM[k] := \sum_{j \in Q_K} W[j]; \quad G[k] := |\bigcup_{KG_i \in Q_K} KG_i|;$$

$$\underset{k=1,...,K}{Max} RM[k] \le L_t \qquad \underset{k=1,...,K}{Max} G[k] \le C_t.$$

==========

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 *Weight-Knapsack* problem, which can be proved to be NP-hard.

Note that the problem has multiple objectives, which cannot be handled by conventional greedy methods. In what follows, we first develop two heuristic algorithms with each taking one of the two objectives as a constraint and optimize the other. Then, the two algorithms are run to get two solutions, respectively, and the better one is chosen finally.

Capacity First Algorithm (CFA): The objective $\underset{k=1,...,K}{Max} RM[k]$ is regarded as a constraint. In this algorithm, the Key-ID groups with relatively more rules will be distributed first. In each round, the current Key-ID group will be assigned to the TCAM with the least number of rules under the load constraint.

==========

I) Sort $\{i, \ i = 1,2,...,2^P\}$ in decreasing order of $|KG_i|$, and record the result as $\{Kid[1], Kid[2],..., Kid[2^P]\}$;

II) for *i* from 1 to 2^P **do**

Sort $\{k, k = 1,...,K\}$ in increasing order of $RM[k]$, and record as $\{Sc[1], Sc[2],..., Sc[K]\}$;

for *k* from 1 to *K* **do**

if $RM[Sc[k]] + W[Kid[i]] \le L_t$

then $Q_{Sc[k]} = Q_{Sc[k]} \bigcup KG_{Kid[i]}$

$G[Sc[k]] = |Q_{Sc[k]}|$;

$RM[Sc[k]] = RM[Sc[k]] + W[Kid[i]]$;

break;

III) Output $\{Q_k, k = 1,...,K\}$ and $\{RM[k], k = 1,...,K\}$.

==========

Load First Algorithm (LFA): In this algorithm, the capacity objective is regarded as a constraint. The Key-ID groups with relatively larger traffic load ratio will be assigned to TCAM first, and the TCAM chips with lower load are chosen.

==========

I) Sort $\{i, \ i = 1,2,...,2^P\}$ in decreasing order of $W[i]$, and record the result as $\{Kid[1], Kid[2],..., Kid[2^P]\}$;

II) for *i* from 1 to 2^P **do**

Sort $\{k, k = 1,...,K\}$ in increasing order of $RM[k]$, and record as $\{Sc[1], Sc[2],..., Sc[K]\}$;

for *k* from 1 to *K* **do**

if $|Q_{Sc[k]} \bigcup KG_{Kid[i]}| \le C_t$

then $Q_{Sc[k]} = Q_{Sc[k]} \bigcup KG_{Kid[i]}$

$G[Sc[k]] = |Q_{Sc[k]}|$;

$RM[Sc[k]] = RM[Sc[k]] + W[Kid[i]]$;

break;

III) Output $\{Q_k, k = 1,...,K\}$ and $\{RM[k], k = 1,...,K\}$.

==========

The Distributed Table Construction Scheme: The two algorithms may not find a feasible solution with a given L_t value. Hence, they are iteratively run by relaxing L_t in each iteration until a feasible solution is found. In a given iteration, if only one of the two algorithms finds a feasible solution, this solution would be the final one. If both algorithms find feasible solutions, one of them chosen according to the following rules:

Suppose that G_A and RM_A are the two objectives given by algorithm *A* (CFA or LFA), while G_B and RM_B are the two objectives given by a different algorithm *B* (LFA or CFA).

1) If $G_A < G_B$, and $RM_A < RM_B$, we choose the solution given by algorithm *A*;

2) If $G_A < G_B$, but $RM_A > RM_B$, we choose the solution given by algorithm *A* when $RM_A < 2/K$ (the reason will be revealed shortly in Section III.D), otherwise we choose the solution given by algorithm *B*.

The corresponding processing flow is depicted in Fig.3.

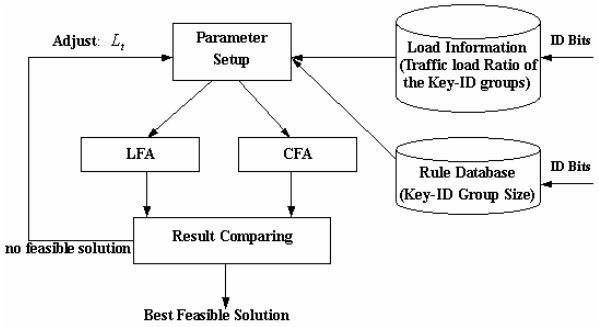

Fig. 3 Distributed Table Construction Flow.

We still use the rule database set #5 as an example. Suppose that the traffic load distribution among the Key-ID groups is as depicted in Fig. 4, which is selected intentionally to have 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 sub-section. Given the constraint C_t=600 and L_t=30%, the results for K=4 and K=5 are shown in Tables II and III, respectively.

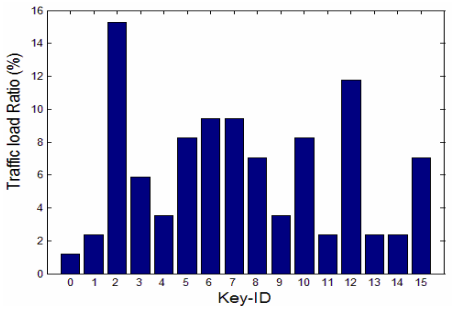

Fig. 4 Traffic Load Distribution among the Key-ID (Key-ID groups).

For K=5, CFA produces a better result (both objectives are better) than that of LFA, as shown in TABLE II. We find that the numbers of rules assigned to different TCAMs are very close to one another. The Distributed storage Expansion Ratio (DER) is 1.51, which means that only about 50% more TCAM entries are required (note that 200% (at K=2) or more are required in the case when the rule table is duplicated and assigned to each TCAM). The maximum traffic load ratio is 29.4%<2/K=40%. As we shall see soon, using the load-balancing schemes proposed in Section III.D, this kind of traffic distribution can be perfectly balanced.

For K=4, LFA instead produces a better result than that of CFA and the maximum and minimum traffic load ratios are 25.9% and 23.5%, respectively, very close to a perfectly balanced load.

TABLE II When K=5, CFA gives the best result. No iteration is needed.

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	18.8
#2	8(1000)	7(0111)	4(0100)		40	439	20.0
#3	15(1111)	10(1010)	14(1110)	12(1100)	40	489	29.4
#4	9(1001)	3(0011)	13(1101)		36	445	11.8
#5	5(0101)	6(0110)	1(0001)		36	494	20.0
Distributed Storage Expansion Ratio (*DER*)						2345/1550=1.51	

TABLE III When K=4, LFA gives the best result. No iteration is needed.

TCAM	Key-ID Groups (Table Contents)				Number of Rule-ID Groups	Number of Rules	Traffic Load Ratio%
#1	2(0010)	15(1111)	13(1101)	0(0000)	45	591	25.9
#2	12(1100)	8(1000)	9(1001)	11(1011)	40	532	24.7
#3	6(0110)	5(0101)	3(0011)	14(1110)	46	586	25.9
#4	7(0111)	10(1010)	4(0100)	1(0001)	50	596	23.5
Distributed Storage Expansion Ratio (*DER*)						2304/1550=1.48	

C. Solutions for Range Matching

Range matching is a critical issue for effective use of TCAM for PC. The real word databases in [10] showed the TCAM storage efficiency can be as low as 16% due to the existence of a large number of rules with ranges. We apply our earlier proposed Dynamic Range Encoding Scheme (DRES) [15] to distributed TCAMs, in order 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 sub-field with ranges. An encoded range is mapped to a code vector implemented using the free bits, and the corresponding subfield is wildcarded. Hence, a rule with encoded ranges can be implemented in 1 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 search Key Encoding (KE). Accordingly, the PC process in a TCAM with range encoding includes two steps: search KE and Rule Matching (RM). DRES uses the TCAM coprocessor itself for KE to achieve wire-speed PC performance. If the encoded ranges come from S sub-fields, S separate range tables are needed for search KE. The S range tables as well as the rule table can be allocated in the same or different TCAMs. The KE involves S sub-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 PC with range encoding requires S range table lookups for KE and 1 RM lookup.

For typical 104-bit five-tuple rules, ranges only appear in the source and destination port subfields, and hence only 2 range tables are needed. For a TCAM with 64-bit slot size, each rule takes 2 slots, and leaves 24 free bits for range encoding. Each RM takes 2 TCAM lookups (each slot takes 1 lookup). A range coming from the source/destination port sub-field takes 1 slot in a range table and hence incurring 1 TCAM lookup for each range table matching. In summary, there are a total number of 4 TCAM lookups per PC. With a 100 MHz TCAM at 100 million lookups per second, DRES can barely support OC192 (i.e. 25 Mpps) wire-speed performance. The distributed TCAM scheme that exploits CLP to increase the TCAM lookup performance is needed to support line rates higher than OC192. The following sections present the details on how to incorporate DRES into the proposed distributed solution.

D. Efficient Load-Balancing Schemes

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 a very efficient

way to dynamically balance the PC traffic in response to traffic pattern changes. The key idea is to duplicate range encoding tables to all the TCAMs and hence allow a KE to be performed using any one of the TCAMs to dynamically balance the load. Since the size of the range tables are small, e.g., no more than 15 entries for all the 5 real-world databases, duplicating range tables to all the TCAMs does not impose distinct overhead.

We design two algorithms for dynamic RE. First, we define some mathematical terms. Let $D[k]$ be the overall traffic load ratio assigned to TCAM #k (k=1,2,…K), which includes two parts, i.e., the KE traffic load ratio and the RM traffic load ratio, with each contributes 50% of the load, according to Section III.C.

Let $KE[k]$ and $RM[k]$ be the KE and RM traffic ratio allocated to TCAM #k, (k= 1,2,…,K_), respectively. Note that $RM[k]$ is determined by the Distributed Table Construction process (refer to Section III.B).

Let $A[i,k]$, $A[i,k] \geq 0, i,k = 1,..., K, \sum_i A[i,k] = 1$, be the *Adjustment Factor Matrix*, which is defined as the percentage (ratio) of the KE tasks allocated to TCAM #i, for the corresponding RM tasks which are performed in TCAM #k. Then the dynamic load balancing problem is formulated as follows:

==========

To decide $A[i,k], i,k = 1,..., K$

Minimize:

$$\underset{k=1,...,K}{Max}\, D[k] - \underset{k=1,...,K}{Min}\, D[k].$$

Subject to:

$$D[k] = 0.5 \times KE[k] + 0.5 \times RM[k], k = 1,..., K;$$

$$KE[i] = \sum_{k=1,...,K} A[i,k] \times RM[k], i = 1,..., K.$$

==========

The following two algorithms are proposed to solve the above problem.

Stagger Round Robin (SRR): The idea is to allocate the KE tasks of the incoming packets whose RM tasks are performed in a specific TCAM to other TCAM chips in a Round-Robin fashion. Mathematically, this means that:

$A[k,k] = 0$ and $A[i,k] = 1/(K-1), , i \neq k, i,k = 1,..., K$.

We then have,

$D[k]=0.5\times(RM[1]+...+RM[k-1]+RM[k+1]+...+RM[K])/(K-1)+0.5\times RM[k]$

$k = 1,..., K$; therefore

$$\underset{k=1,...,K}{Max}\, D[k] - \underset{k=1,...,K}{Min}\, D[k] = 0.5 \times (\underset{k=1,...,K}{Max}\, RM[k] - \underset{k=1,...,K}{Min}\, RM[k]) \times (K-2)/(K-1).$$

Comments: In the case when K=2, the objective is a constant "0". This means that no matter how large the variance of the RM load ratios among all the TCAM chips is, SRR can always perfectly balance the overall traffic load. Since $0.5 \times (K-2)/(K-1) < 0.5$, it means in any case, SRR can always reduce the variance of the overall load ratio to less than half of that of the RM tasks.

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 TCAM chip. Whenever a packet arrives, the corresponding KE task is assigned to the TCAM who has the smallest counter value.

In this case, the values of $A[k,i]$ are not fixed. The expression of $D[k]$ is given by:

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

Note that $0 \leq A[k,i] \leq 1, i = 1,..., K$, we have

$0.5 \times RM[k] \leq D[k] \leq 1, k = 1,..., K$.

Taking $A[i,k]$ as tunable parameters, it is straightforward that the equations:

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

$k = 1,..., K$,

must have feasible solutions when $0.5 \times RM[k] \leq 1/K$

i.e., $RM[k] \leq 2/K$, $k = 1,..., K$.

This means that if the conditions: $RM[k] \leq 2/K$, $k = 1,..., K$, are all satisfied, the overall traffic load ratio can be perfectly balanced (the objective value is 0) in the presence of traffic pattern changes.

Comments: The overall traffic load can be perfectly balanced when $RM[k] \leq 2/K, k = 1,..., K$, are satisfied, which makes FA a very efficient solution when compared with SRR. However, FA incurs more implementation cost due to the need of a counter for each TCAM chip.

Further discussions on the performance of SRR and FA are presented in Section V.

IV. Implementation of the DPPC-RE Scheme

The detailed implementation of the DPPC-RE mechanism is depicted in Fig.5. Beside the TCAM chips and the associated SRAMs to accommodate the match conditions and the associated actions, three major additional components are included in co-operating 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 these components in details.

A. The Distributor

This component is actually a scheduler. It partitions the PC traffic among the TCAM chips. More specifically, it performs three major tasks. First, it extracts the Key-ID from the 5-tuple received from a network processing unit (NPU). The Key-ID is used as an identifier to dispatch the RM keys to the associated TCAM. The 5-tuple is pushed into the RM FIFO queue of the corresponding TCAM (Solid arrows in Fig. 5).

Second, the distributor distributes the KE traffic among the TCAM chips, based on either the FA or SRR algorithm. The corresponding information, i.e., the SPORT and DPORT are pushed into the KE FIFO of the TCAM selected (dashed arrows in Fig.5).

Third, the distributor maintains K Serial Numbers (S/Ns) or

S/N counters, one for each TCAM. An S/N is used to identify each incoming packet (or more precisely, each incoming five-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 RM TCAM. The format of the Tag is depicted in Fig.6(a).

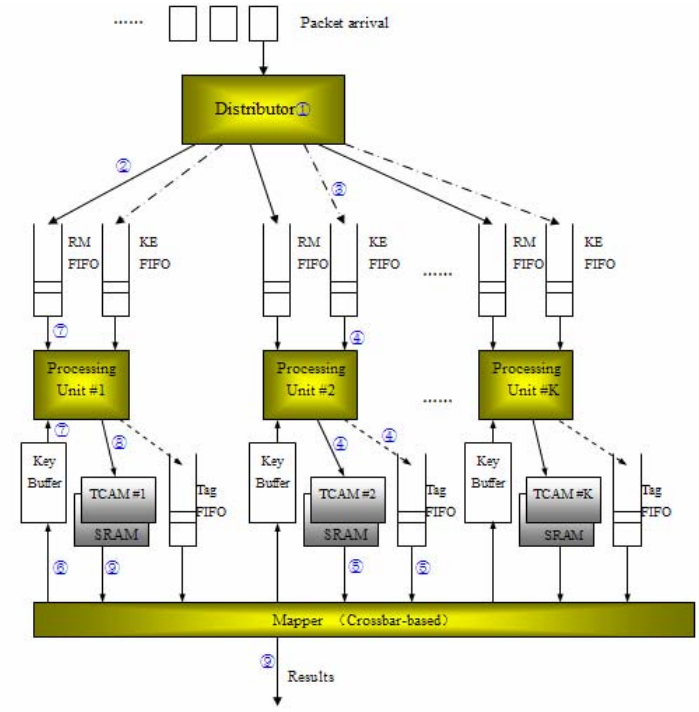

Fig. 5 DPPC-RE mechanism.

As we shall explain shortly, the tag is used by Mapper to return the KE results back to the correct TCAM and to allow the PU for that TCAM to establish the association of these results with the corresponding five-tuple in the RM queue.

S/N(5)	CAMID(3)

(a) Tag Format

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

(b) RM Buffer Format

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

(c) KE Buffer Format

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

(d) Key Buffer Format

Fig. 6 Format of Tag, RM FIFO, KE FIFO and Key Buffer.

B. RM FIFO, KE FIFO, Key Buffer, and Tag FIFO

A RM FIFO is a small FIFO queue where the information for RM of the incoming packets is held. The format of each unit in the RM FIFO is given in Fig.6(b). (The numbers in the brackets indicate the number of memory bits needed for the sub-fields).

A KE FIFO is a small FIFO queue where the information used for KE is held. The format of each unit in the KE FIFO is given in Fig.6(c).

Differing 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 KE (encoded bit vectors of the range sub-fields) are held. The size of a Key Buffer equals to the size of the corresponding RM FIFO, with one unit in the Key Buffer corresponds to one unit in the RM FIFO. The format of each unit is given in Fig.6(d). 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.

C. The Processing Unit

Each TCAM is associated with a Processing Unit (PU). The functions of a PU are to (a) schedule the RM and KE tasks assigned to the corresponding TCAM, aiming at maximizing the utilization of the corresponding TCAM; (b) ensure that the results of the incoming packets assigned to this TCAM are returned in order. In what follows, we elaborate on these two functions.

(a) Scheduling between RM and KE tasks: Note that, for any given packet, the RM operation cannot take place until the KE results are returned. Hence, it is apparent that the units in a RM FIFO would wait for a longer time than the units in a KE FIFO. For this reason, RM tasks should be assigned higher priority than KE tasks. However, our analysis (not given here due to the page limitation) indicates that a strict-sense priority scheduler may lead to non-deterministically large processing delay. So we introduce a Weighted-Round-Robin scheme in the PU design. More specifically, each type of tasks gain higher priority in turn based an asymmetrical Round-Robin mechanism. In other words, the KE tasks will gain higher priority for one turn (one turn represents 2 TCAM accesses, for either a RM operation or two successive KE operations) after n turns with the higher priority assigned to RM tasks. Here n is defined as the Round-Robin Ratio (RRR).

(b) Ordered Processing: Apparently, the order of the returned PC results from a specific TCAM is determined by the processing order of the RM operation. Since a RM buffer is a FIFO queue, the PC results can still be returned in the same order as the packet arrivals, although the KE tasks of the packets may not be processed in their original sequence [iv]. As a result, if the KE result for a given RM unit returns earlier than those units in front of it, this RM unit cannot be executed.

Specifically, the PU for a given TCAM maintains a pointer points to the position in the Key Buffer that contains the KE 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 RM FIFO. In each TCAM cycle, PU queries the valid bit of the

[iv] This is because the KE tasks whose RM is processed in a specific TCAM may be assigned to different TCAMs to be processed based on the FA or SRR algorithms.

position that the pointer points to in the Key Buffer. If the bit is set, meaning that the KE result is ready, and it is RM's turn for execution, PU reads the KE results out from the Key Buffer and the 5-tuple information out from the RM FIFO queue, and launches the RM 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.

D. The Mapper

The function of this component is to manage the result returning process of the TCAM chips. According to the processing flow of a PC operation, the mapper has to handle three types of results, i.e., the KE Phase-I results (for the SPORT sub-field), the KE-Phase-II results (for the DPORT sub-field), and the RM results. The type of the result is encoded in the result itself.

If the result from any TCAM is a RM result (which is decoded from the result itself), the mapper returns it to the NPU directly. If it is a KE-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 KE-Phase II result, the mapper uses the tag information from the Tag FIFO to determine: 1) which Key Buffer (according to the CAMID segment) should this result be returned to, and 2) which unit in the Key Buffer (according to the S/N segment) should this result be written into. Finally the mapper combines the 2 results (of Phase I and II) into one and returns it.

E. An Example of the PC 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 II (in Section III.B). 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. 5):

①: The 4-bit Key-ID "0010" is extracted by Distributor.

②: According to the distributed rule table given by TABLE II, Key-ID group "0010" is stored in TCAM#1. Suppose that the current S/N value of TCAM#1 is "5", then the CAMID "001" and S/N are combined into the Tag with value "00110(5+1)". 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 5 KE FIFOs are 2,0,1,1, and 1, respectively. According to the FA algorithm, the KE operation of packet *P0* is to be performed in TCAM#2. Then the two range sub-fields <15535, 80>, together with the Tag, are pushed into the KE FIFO associated with TCAM#2.

④: Suppose that now it is KE's turn or no RM task is ready for execution, PU#2 pops out 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 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 6th 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 RM's turn, PU#1 probes the valid bit of the 6th 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 6th unit of the Key Buffer (the encoded key of the two ranges), and then launches a RM operation in TCAM#1. Meanwhile, the valid bit of the 6th unit in the Key Buffer is reset and the pointer of PU#1 is incremented by one and points to the 7th unit.

⑨: When Mapper receives the RM result, it returns it back to the NPU, completing the whole PC process cycle for packet *P0*.

V. EXPERIMENTAL RESULTS

A. Simulation Results

Simulation Setup: Traffic Pattern: Poisson Arrival process; Buffer Size: RM FIFO=8; Key Buffer=8, KE FIFO=4, Round-Robin-Ratio=3; Traffic Load Distribution among Key-ID groups: given in Fig 2.

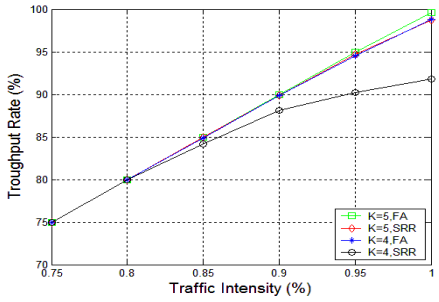

Fig. 7 Simulation results (Throughput).

Throughput Performance: The simulation results are given in Fig. 7. One can see that at K=5, the OC-768 throughput is guaranteed even when the system is heavily loaded (traffic intensity tends to 100%), whether FA or SRR algorithm is adopted. This is mainly because the theoretic throughput upper bound at K=5 (5*100M/4=125Mpps) is 1.25 times of the OC768 maximum packet rate (100Mpps). In contrast, at K=4, the throughput falls short of the wire-speed when SRR is used, while FA performs fairly well, indicating that FA has better load-balancing capability than SRR.

Delay Performance: According to the processing flow of the DPPC-RE scheme, the minimum delay for each PC is 10 TCAM cycles (5 for RM and 5 for KE). In general, however, additional cycles are needed for a PC because of the queuing effect. We focus on the performance when the system is heavily loaded. Fig. 8 shows the delay distribution for the back-to-back

mode, i.e., when packets arrive back-to-back (Traffic intensity =100%).

We note that the average delay are reasonably small except for the case at K=4 and when SRR is adopted (avg.delay>20 TCAM cycles). In this case, when the offered load reaches the theoretical limit (i.e., 100 Mpps), a large number of packets are dropped due to SRR's inability to effectively balance the load.

The delay distributions for the cases using FA (K=4 or 5) are much more concentrated than those using SRR, suggesting that FA offer much smaller and more deterministic delay performance than SRR. Note that more deterministic delay performance results in less buffer/cache requirements and lower implementation complexity for the TCAM Classifier as well as other components in the fast data path.

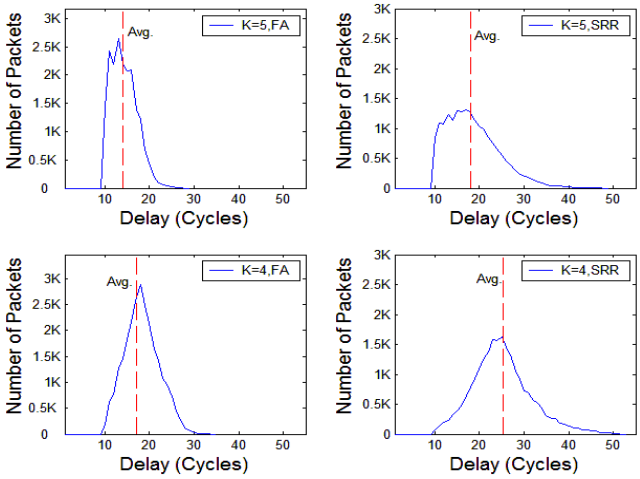

Fig. 8. Delay Distributions of the four simulations.

Change of Traffic Pattern: 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 (traffic intensity=100%).

The traffic pattern depicted in Fig. 2 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 are given in Table IV.

TABLE IV Simulation setups.

Case	Number of TCAM	FA/SRR	Table constructed from	Traffic change to
I	5	FA	Pattern I	Pattern II
II	5	FA	Pattern II	Pattern I
III	4	FA	Pattern I	Pattern II
IV	4	FA	Pattern II	Pattern I
V	4	SRR	Pattern II	Pattern I
VI	4	SRR	Pattern I	Pattern II

The results are given in Table V. We find that although the traffic pattern changes significantly, the throughput performance just decreases slightly[v] (<1%) in all the cases when FA are adopted. This means that FA excels in adapting to traffic pattern changes. The performance of SRR is a bit worse (>4% in Case V). Overall, we may conclude that the DPPC-RE scheme copes with the changes of traffic pattern well.

TABLE V Throughput ratios in the presence of traffic pattern changes.

Case	Before (%)	After (%)
I	99.76	99.96
II	100	99.63
III	99.24	98.68
IV	98.99	98.07
V	92.76	88.39
VI	93.38	91.71

B. Comparison with other schemes

Since each PC operation needs at least 2 TCAM accesses, as mentioned in Section III, a single 100MHz TCAM chip cannot provide OC768 wire-speed (100Mpps) PC. 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 adopting Key Encoding or not, there would be four different possible schemes. They are:

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

2) *Duplicate Table + Key Encoding*: Four 100MHz TCAM chips should be used in parallel to achieve 100Mpps, with each containing an encoded rule table (with N_e entries). A total number of $K \times N_e$ TCAM entries are required. It also offers lossly, deterministic performance;

3) *Distributed Storage + Key Encoding (DPPC-RE)*: Four or five 100MHz 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 100MHz 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 un-deterministic and massive loss may occur when the system is heavily loaded or 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 five real-world databases based on these four schemes. The results are given in TABLE VI.

Apparently Distributed+KE, or DPPC-RE significantly outperforms all the other three schemes in terms of the TCAM storage efficiency. Moreover, with only a slight increase of ER for K=5, compared with K=4, OC-768 wire-speed PC throughput performance can be guaranteed for Distributed +KE

[v] In Case I, the throughput even increase, 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

(K=5) case.

TABLE VI Comparison of the Expansion Ratio.

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 when paralleled to Support OC768					
Duplicate+NoKE(K=2)	6.80	6.04	5.98	12.40	2.82
Duplicate+KE(K=4)	4.06	4.09	4.08	4.11	4.02
Distributed+KE (K=5)	1.59	1.70	1.82	2.19	1.53
Distributed+KE (K=4)	**1.44**	**1.66**	**1.76**	**2.14**	**1.51**
Distributed+NoKE(K=2)	5.18	4.46	3.43	7.81	1.69

Obviously, DPPC-RE exploits the tradeoff between deterministic performance and high statistical throughput performance, while the schemes with table duplication gains high, deterministic performance at significant memory cost. Because 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, it is an attractive solution.

VI. Discussion and 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 high performance and low cost packet classifier for next generation multi-gigabit router interfaces.

In this paper, a distributed parallel packet classification scheme with range encoding (DPPC-RE) is proposed to achieve OC768 (40 Gbps) wire-speed packet classification with minimum TCAM cost. DPPC-RE includes a rule partition algorithm to distribute rules into different TCAM chips with minimum redundancy, and a heuristic algorithm to balance the traffic load and storage demand among all TCAMs. The implementation details and a comprehensive performance evaluation are also presented.

A key issue that has not been addressed in this paper is how to update the rule and range tables with minimum impact on the packet classification process. The consistent policy table update algorithm (CoPTUA) [19] proposed by two of the authors allows the TCAM policy rule and range table to be updated without impacting the packet classification process. CoPTUA can be easily incorporated into the proposed scheme to eliminate the performance impact of rule and range table update. However, this issue is not discussed in this paper, due to space limitation.

VII. Acknowledgement

The authors would like to express their gratitude to Professor Jonathan Turner and Mr. David Taylor from Washington University in St. Louis for kindly sharing their real-world databases and the related statistics with them.

VIII. Reference

[1] V. Srinivasan, George Varghese, Subhash Suri, and Marcel Waldvogel, "Fast and Scalable Layer Four Switching", *Proc. of ACM SIGCOMM* , 1998.

[2] T.V. Lakshman, and Dimitrios Stiliadis, "High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching", *Proc. of ACM SIGCOMM* ,1998.

[3] M. Buddhikot, S. Suri, and M. Waldvogel, "Space Decomposition Techniques For Fast Layer-4 Switching", *Protocols for High Speed Networks IV (Proceedings of PfHSN '99)*.

[4] A. Feldmann, and S. Muthukrishnan, "Tradeoffs for Packet Classification", *Proc. of IEEE INFOCOM*, 2000.

[5] F.Baboescu, S.Singh, G.Varghese, "Packet Classification for Core Routers: Is there an alternative to CAMs?", *Proc. of IEEE INFOCOM*, San Francisco USA, 2003.

[6] Pankaj Gupta, and Nick McKeown, "Packet Classification on Multiple Fields", *Proc. of ACM SIGCOMM,* 1999.

[7] Pankaj Gupta, and Nick McKeown, "Packet Classification using Hierarchical Intelligent Cuttings", *IEEE Micro Magazine*, Vol. 20, No. 1, pp 34-41, January- February 2000.

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

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

[10] E. Spitznagel, D. Taylor, J. Turner, "Packet Classification Using Extended TCAMs", In *Proceedings of International Conference of Network Protocol (ICNP)*, September 2003.

[11] T. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using ef_cient multi-dimensional range matching," *ACM SIGCOMM Computer Communication Review*, Vol. 28, No. 4, pp. 203-214, October 1998.

[12] H. Liu, "Efcient Mapping of Range Classier into Ternary CAM", *Proc. of the 10th Symposium on High Performance Interconnects (hoti'02)*, August, 2002.

[13] J. van Lunteren and A.P.J. Engbersen, "Dynamic multi-field packet classication", *Proc. of the IEEE Global Telecommunications Conference Globecom'02*, pp. 2215 -2219, November 2002.

[14] J. van Lunteren and A.P.J. Engbersen, "Fast and Scalable Packet Classication", *IEEE Journal of Selected Areas in Communications*, Vol. 21, No. 4, pp560-571, May 2003.

[15] H. Che, Z. Wang, K. Zheng, and B. Liu, "DRES: Dynamic Range Encoding Scheme for TCAM Coprocessors," submitted to *IEEE Transactions on Computers.* It is also available online at: http://crewman.uta.edu/~zwang/dres.pdf.

[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. of IEEE INFOCOM*, April, 2004.

[17] John L. Hennessy, and David A. Patterson, Computer Architecture: A Quantitative Approach, *The China Machine Press*, ISBN 7-111-10921-X, pp. 12 and pp.390-391.

[18] Please see Acknowledgement.

[19] Z, Wang, H. Che, M. Kumar and S. K Das, CoPTUA: Consistent Policy Table Update Algorithm for TCAM without Locking, *IEEE Transactions on Computers,* 53(12) 1602-1614, 2004.