IPv6-Oriented 4×OC-768 Packet Classification with Deriving-Merging Partition and Field-Variable Encoding Algorithm[i]

Xin Zhang[1], Bin Liu[2], Wei Li[2], Ying Xi[2], David Bermingham[3], Xiaojun Wang[3]
[1]Department of Automation, Tsinghua University, Beijing, P.R.China 100084
[2]Department of Computer Science, Tsinghua University, Beijing, P.R.China 100084
[3]School of Electronic Engineering, Dublin City University, Dublin 9, Ireland
tsinghuazhangxin@yahoo.com.cn, liub@mail.tsinghua.edu.cn, li-wei03@mails.tsinghua.edu.cn

***Abstract*—Packet Classification serves as a plinth for many newly emerging network applications. Most of the previous packet classification schemes are IPv4-oriented, and some of them have achieved high throughput with chip-level parallelism of Ternary Content Addressable Memories (TCAM). However, due to their inefficient utilization of TCAM resources, further upgrade incurs prohibitive hardware costs. As IPv6 will dominate the Next Generation Internet, IPv6-oriented packet classification is of increasing importance. In this paper, we propose a packet classification scheme geared towards IPv6. This scheme incorporates efficient and flexible algorithms for parallelism and distributed storing, which provides an unprecedentedly high throughput with relatively low storage costs. Our scheme also integrates delicate parallel encoding algorithms to maximize the TCAM utilization and increase its throughput. Using commercially available TCAM, the scheme is able to classify 266 million IPv6 packets per second (Mpps), matching 4×OC-768 (160 Gbps) line rate.**

Key words—Packet Classification, Encoding, IPv6, TCAM

I. INTRODUCTION

The development of Internet applications necessitates the router's ability to support functions such as QoS (Quality of Service), traffic measurement, network intrusion detection, congestion control and firewall etc., all of which rely on the implementation of packet classification. The throughput and robustness of packet classification exercise dramatic influence on network performances. Packet classification refers to querying all the *rules* in the *policy table* to find the matched one with the highest priority against the incoming s*earch key* extracted from the IP packet header. Both a rule and a search key generally consist of multiple fields. For either IPv4 or IPv6 [2, 10], a rule is typically comprised of five fields: source and destination IP addresses (SIP and DIP) in the form of prefixes; source and destination ports (SP and DP) presented as ranges such as *[1:10]*, *greater than 1024* etc.; and protocol specified by exact numbers. Correspondingly, the matching process involves the longest prefix matching (LPM), exact matching and range matching which refers to finding the port range(s) that a port number contained in the search key belongs to. Due to its complexity, packet classification has become one of the primary concerns in network designs.

TCAM-based schemes present remarkable merits among existing solutions. TCAM, a specialized memory device linearly organized in *slot*s with fixed size, is inherently suited to LPM by allowing a "don't care" state ('*') to be stored in memory cells in addition to '0's and '1's. Besides, a TCAM returns the matching result within a single memory access by querying all its slots in parallel within one operation. Though TCAM-based schemes can thus achieve high throughput, further throughput upgrade for ultra-high end applications is still severely restricted by the TCAM accessing speed. Current TCAM can work at a speed up to 133MHz[ii], however it cannot provide sufficient accessing throughput for IPv6 classification at ultra high line rates, because multiple TCAM cycles may be needed for a single matching due to the increased length of IPv6 search keys. To overcome the accessing speed limitation of a single TCAM, chip-level parallelism among TCAMs is exploited in [16] to match the OC-768 line rate. Instead of naively duplicating the complete policy table in all TCAMs, [16] employs a distributed storing strategy, i.e., it partitions the original policy table into multiple s*ub groups*, each of which is smaller than the original one and each sub group is stored in only one TCAM. After our extensive investigation, [16] is one of the few schemes with distinctly superior performances to others in related topics (thus we mainly take it for comparison), but the scheme still fails to solve problems.

The key objectives and obstacles for designing an effective and scalable for IPv6 packet classification engine based on parallel TCAMs lie in that: 1) the much longer IPv6 rule (a typical IPv4 5-field rule is 104-bit long while the IPv6 one is 296-bit long) poses great difficulties in storing and matching, since most established algorithms need much longer time for the larger rule/search key matching, resulting in a sharp decrease of throughput. Also, the dire destitution of IPv6 real traces and policy tables in current stage makes most algorithms in [16] impractical, as they particularly rely on the real-word observations; 2) the number of parallel TCAMs cannot be freely increased for an arbitrarily high throughput, considering the excessive power consumption, expensive TCAM device costs and the system complexity. The most economical and feasible method is to maximize the TCAM utilization, i.e., to engage as much portion of the TCAM cycles as possible to the classification matching tasks. However, half of the TCAM cycles in [16] are occupied by other tasks (i.e., encoding the search keys); 3) to enhance the TCAM accessing and storing efficiency, search key encoding is essential as explained later

This research is supported by the NSFC under Grant No. 60373007 and No. 60573121; China-Ireland Science and Technology Collaboration Research Fund (CI-2003-02) and Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048); 985 Fund of Tsinghua University(No. JCpy2005054)

[ii] 133MHz equals to 266MSPS since by certain new feature of TCAM products, two parallel searches in different databases can be performed simultaneously for a given input key. This feature is utilized in this paper and one TCAM cycle consists of two parallel searches.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the Proceedings IEEE Infocom.

in Section II. A search key encoding scheme is expected to meet the ultra high line rate (4×OC-768 in this paper) and be independent of the rule-storing specifics in TCAMs, thus making the system flexible and easy to update. Nevertheless, the encoding scheme in [16] dissipates TCAM accesses; 4) the traffic load (rule-matching load) distribution and storage allocation among TCAMs should be balanced to ensure a deterministic high throughput and economical memory costs, while existing schemes are ineffective for this multi-objective optimization and the real-time traffic-adjusting scheme in [16] reduces the throughput to half and is not universal.

To prepare for the ever-increasing line rates and IPv6 new features, in this paper we present an IPv6-oriented packet classification scheme matching $4\times$ OC-768 line rate with relatively low hardware costs. In light of the aforementioned objectives and obstacles, 1) this scheme fully accounts for the IPv6 new features while maintaining compatibility with IPv4; and most algorithms are independent of characteristics of real policy tables, making our scheme flexible and unrestricted by the lack of IPv6 real-world data; 2) to maximize the TCAM utilization, encoding loads are devolved from TCAMs and a separate encoding module is created by skillfully exploring the parallelism to the largest extent but with minimum storage costs; 3) an improved TCAM-chip-level parallelism is deployed in pursuit of an unprecedentedly high throughput, accompanied by multiple algorithms including the IPv6 rule compression, distributed parallelism and real-time traffic adjustment etc., by which both the minimum costs and high performances are achieved.

The rest of this paper is organized as follows: Section II outlines the key ideas and system architecture; Section III presents detailed algorithms and solutions; Section IV elaborates specifics of the system implementation; Section V and VI derive performance evaluations and experimental results respectively; Section VII finally concludes the paper.

II. OVERVIW OF KEY IDEAS AND SYSTEM ARCHITECTURE

In this paper, we deploy parallelism both in the encoding modules and among TCAMs to realize packet classification at 4×OC-768 line rate. As is depicted in Fig.1, the architecture of our scheme is comprised of the Network Processor (NP) and the external devices, i.e., TCAMs and associated SRAMs. The logics and modules within NP coordinate parallel operations and prepare for the packet classification matching in the external devices. The key modules and ideas are as follows.

A. Coordinator

As a parallel system, the coordination and collaboration are two primary concerns. The coordinator functions to guarantee both the correctness of parallelism and the high performance of the system. Specifically speaking, 1) the incoming packets arrive in ordered sequence, parallel processing within the system naturally tends to trigger sequence disorder as the parallel latencies may differ. Logic [a] in Fig.1 inserts sequence numbers (S/N) to a certain subset of[iii] search keys and their original order is reconstructed in [i] according to their S/Ns; 2) the level of balance of the distributed traffic loads among TCAMs hugely affects the system throughput. In addition to evenly distributing the sub groups to TCAMs with respect to their load ratios, we select appropriate sub groups to be duplicated in all TCAMs for further throughput balancing: the corresponding classification loads, called "Global Traffic" (the others are called "Local Traffic" in this paper), are always assigned to the currently least loaded TCAM. Logic [b] generates a label for each search key indicating which sub group it *belongs to*, i.e., which sub group contains the rule(s) it matches. Also, both the real-time information on TCAM loads and the scheduling decision are maintained and made in [b].

B. Encoding

Assuming prefix specification for packet classification, an *Entry*, consisting of one or more TCAM slots, can store a rule by concatenating the prefixes of all fields. However the ranges in SP and DP, which can not be straightforwardly expressed by ternary strings, must be split into a set of prefix formats, excessively increasing the number of rule entries in TCAMs and reducing the update speed. One pervasive solution is to encode the ranges to '0-1' binary string format and the TCAM entries are efficiently reduced by storing these strings instead. Besides, as the length of an IPv6 rule grows up to 296-bit long, we also encode SIP, DIP and protocol fields to shorter lengths to economically fit a rule into one TCAM slot. Encoded rules are stored in TCAMs of the most economical size while the incoming search keys also need to be encoded in the same manner. Search key encoding is virtually the process of searching for the codes corresponding to the search keys from an encoding table. Not to take up the TCAM cycles, these encoding tasks are performed in the NP's on-chip RAMs. A parallel structure called the "Independent Field Search Module (IFSM)" is used to search for the codes of each field in a search key independently. The *Encoding Unit* (EU) for each field also runs in parallel for an ultra high throughput. The "Field-variable" parallelism within each field is proposed and shown in Fig.1 [d]. Module [c] splits the search keys for independent searching; and [e] reassembles the encoded parts according to their S/Ns and labels generated from [a] and [b].

C. Interface

The interface ensures the correct collaboration between NP and external devices: 1) Logic [f] distributes the encoded search keys among TCAMs according to the S/Ns and labels derived from [a] and [b], i.e. the global traffic is allocated to the least loaded TCAM while the local traffic to the TCAM containing the sub group to which the key belongs; 2) Module [g] buffers the heavy traffic to each TCAM in case the TCAM load ratios are unbalanced.

D. TCAM Matching and Output

The final classification matching is performed in TCAMs and associated SRAMs. Parallel TCAMs store the encoded rules or sub groups in a distributed fashion while an SRAM contains subsequent *Actions* for matching each rule in its related TCAM. Whenever a TCAM finishes a matching, it feedbacks this message through [h] to update S/Ns, labels and TCAM load information. The matching result for global traffic is sent to [i] for buffering and reordering.

In sum, assisted by the simple logics and delicate modules, the whole system with superior parallelism can operate in a

[iii] As explained in Section IV, only the "Global Traffic " needs to be reordered by S/Ns, while the majority of incoming packets can maintain their original orders without additional efforts. Section V.D further shows that the probability of global traffic disorder is rather slight

concerted and efficient way. Also, the storage costs are relatively low as is analyzed later in this paper.

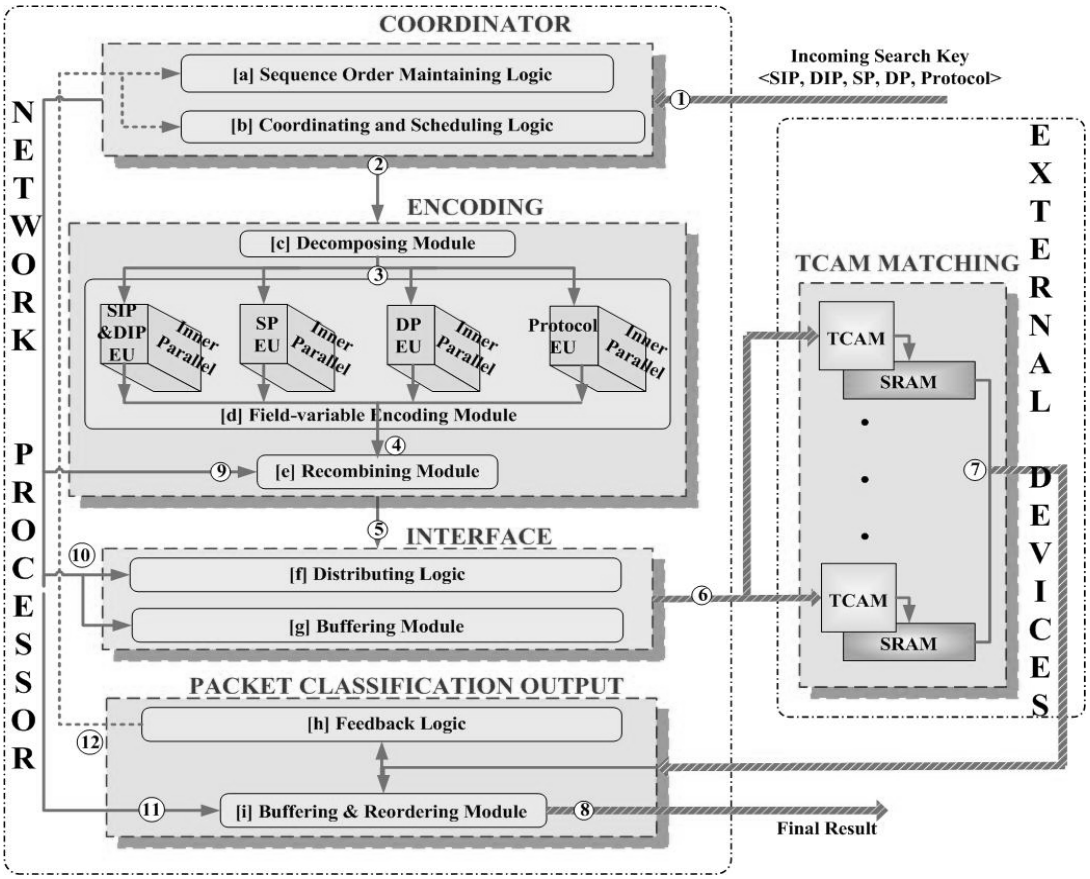


Figure 1. Architecture Overview

III. SOLUTIONS AND ALGORITHMS

A. Compression and Storing Strategy of IPv6 Rules

Normally the five fields: SIP, DIP, SP, DP and protocol are denoted as 5-tuple for simplicity. A 5-tuple IPv6 rule with the length of 296 bits can not be effectively stored in a commercially available TCAM slot[iv]. In this sub section, we present some compression methods for protocol and IPv6 address fields. For SP and DP, an improved layered encoding method based on the P^2C algorithm [2] is proposed to solve the range matching. As a result, the total length of an encoded IPv6 rule is finally compressed to 280 bits. A single TCAM slot with the width of 288 bits (such as Cypress CYNSE70256 TCAM) suffices to store an IPv6 5-tuple rule with the rest 8 bits for other necessary usages, such as to indicate the prefix pair types and so forth. In what follows, we'll give the compression/encoding methods for the 5-tuple rules based on the common properties of real filter sets.

Compressions for Protocol and IPv6 Address Fields:

Since in real filter sets the protocol field is restricted to a small set of values (no more than ten [7]), the original 8 bits in the protocol field can be compressed into 4 bits.

IPv6 addresses fall into three types: multicast, unicast and anycast address. Since the last two have the same address format, we only focus on the former two. 1) For multicast address, the first 8 bits which are all '1's can be directly removed. 2) For unicast address, the minimum network address prefix length is 16bits [7, 8]. Up to now, the number of unique specifications of the first 16 bits is so small to be represented by only 2 bits and there is no tendency to grow fast [9]. For future expansion, we map the first 16 bits to 8bits, supporting up to 256 different specifications of the first 16bits.

Predefined Bits-allocation Range Encoding Solution:

In a 5-tuple rule, SP and DP are represented by five different types of port range specifications: wildcard (WC), ephemeral user port range [1024:65535] (HI), well-known system port range [0:1023] (LO), exact match (EM), and arbitrary range (AR) which is the main cause for range overlapping. We devise a dynamic range encoding method by effectively allocating the code bits in a layered hierarchy denoted by *1/m/n/…*, where the symbol "/" indicates a layer; *m* and *n* represent the number of bits allotted to corresponding layers. First, we give the principles for the predefined bits-allocation: 1) assign *t* bits in total for the range coding; 2) during the initialization, set at least 3 layers; 3) the first layer has one bit to represent HI and LO which obviously overlap with EM and AR; 4) assign *m* bits to the second layer. All the codes for EMs without mutually overlapping are allotted in this layer; 5) the third layer with *n* bits is for the majority of ARs. The remaining layers (if necessary) are for various ARs which may overlap with each other and can not be set in the

[iv] Currently the available slot length configurations are: 36/72/144/288/576/(288+32*N*), where *N* is a subset of natural numbers. ([4,5])

same layer. New layers can be added and existing layers can be removed dynamically one at a time when necessary; 6) all unassigned bits must be initially set to '0'.

Because ARs may overlap one another, according to [2]: we first project all distinct AR fields of a filter set onto a number line to get multiple segments represented by S_i (i=1,2…M), where M is the number of segments converted from the original ARs. The codes of these new segments are obtained as the concatenation of codes in all AR layers, with all bits on irrelevant layers set to '0'. In this way, each range specification (i.e., the derived segments but not the original ARs) is assigned a unique code that is directly stored in SRAMs; while the encoded ternary strings stored in TCAMs can be obtained in the similar manner but with all bits on irrelevant layers set to '*'. Thus the ternary strings are in one to one correspondence with the original port range specifications, eliminating the entry expansion and saving the updating time.

To get the characteristics of real filter sets for determining the values of t, m and n, we use Classbench with a Filter Set Generator [10] to get the port range statistics (though it is just for IPv4, we can still safely use these statistics, because the port field is not on IP network layer but the transportation layer). Our analysis and experimental results show that the total number of bits needed for encoding SP and DP are no more than 7 and 12 respectively. Consequently it is future-safe to pre-assign 12bits and 24bits to SP and DP respectively. Studies have found that the number of unique port range specifications is significantly less than that of filters in the filter set [1]. From the filter statistics in [7], the numbers of unique EMs for SP and DP are at most 27 and 183 respectively; and the numbers of unique ARs are even less, with 5 and 50 for SP and DP respectively. Therefore, we can make initial assignments for SP and DP as 1/6/2 and 1/9/6/2 respectively, which can be gradually extended to 1/6/2/2/1 and 1/9/6/2/2/2/2 by dynamically adding new layers.

Based on the above discussions, a 296-bit key/rule can thus be compressed to a 280-bit one, fully utilizing the 288-bit TCAM slot with the remaining 8 bits for other necessary usage; and a key is input into TCAMs within two TCAM cycles.

B. Field-variable Parallel Encoding Scheme

We expect the encoding scheme to be independent of the rule-storing particulars in TCAMs and match the 4×OC-768 line rate. Correspondingly, NP's on-chip RAMs are used for encoding tasks and the parallelism at three different levels is originally exploited as described below.

Independent Field Search Module (IFSM):

Level 1: We first use parallel structure for range encoding of SP&DP: HI and LO are searched in sequence, and in parallel with AR and EM. These searches produce three intermediate code strings as matching results of the three types: HI/LO, AR and EM (a string of all '0's is returned when there is no match for that type). Then we concatenate the three strings to get the code for the search key. HI and LO are relatively easy to identify. The searching for EM codes utilizes a hash table structure by using the lower k bits of the port number as its hash index. Analysis of statistics indicates that k=5 or 6 for SP and DP respectively can well proportion the set of EM range specifications, and the code for an EM can be obtained within one RAM cycle. We use a T-way search segment tree [12] to perform AR search. Implementing this data structure in SRAM with one node stored in an SRAM cell requires $Log_T M$ RAM search time and $2M/T$ RAM update time, with a space of $(T\times d_I+(T\text{-}1)\times d)\times(M\text{-}1)/(T\text{-}1)$, where d and d_I are bit-widths of port number and index respectively. Setting T=5, the search time is at most six RAM cycles.

Level 2: We use the parallel structure IFSM to simultaneously search for the codes of SIP&DIP, SP, DP and Protocol. By simply using the protocol value as addresses to access the mapping table stored in a 12-bit wide RAM to obtain the corresponding codes, only one RAM cycle is enough for the protocol encoding. For SIP and DIP respectively, the lower k (e.g., k = 8) of the first 16 bits of the IPv6 address are used as hash Index to probe the hash table stored in a 64-bit wide RAM. In this way searching for an IPv6 address code requires only one RAM cycle. The searching time and storage requirements of a single IFSM are given in TABLE I.

Level 3: Our target requires much higher encoding rate than a single IFSM could provide. As a solution we further employ parallelism among IFSMs. There are several ways for parallel IFSMs: we could bond an IFSM and a TCAM as a *Pair* and achieve parallelism at pair level (Approach 1); we could also realize parallelism among IFSMs and TCAMs respectively and independently (Approach 2). However the two approaches are obviously inefficient since the processing times consumed by an IFSM and a TCAM, and even the EUs within one IFSM vary widely. In addition, as the entire IFSM has to be duplicated many times, too much redundancy is introduced.

TABLE I. SEARCHING TIME AND STORAGE REQUIREMENT OF AN IFSM

		Time Req. (RAM Cycle)	Storage Req. (RAM Block)	Storage Req. (Kbit)
SIP&DIP		2	4×4K	16
SP	AR	6	8×M512	4
	EM	1	8×M512	4
DP	AR	6	16×M512	8
	EM	2	16×M512	8
Protocol		1	2×M512	1
Total		6	50×M512 4×M4K	41

Field-variable Parallel Encoding Scheme (FPES):

Differing from the two approaches, FPES allows different fields in the same IFSM to employ parallelism to variable extents. The less encoding time a single EU needs, the fewer parallel EUs are used for this field and the less parallel redundancy this field brings. FPES flexibly matches the throughput of parallel TCAMs with great time efficiency and economical hardware expense. The primary ideas of FPES are: 1) within an IFSM different fields employ different numbers of EUs according to their processing speeds; 2) each of the parallel EUs stores a complete field encoding table and the incoming traffic loads are allotted in a *Round Robin* manner.

To match 4×OC-768 line rate (266Mpps for IPv6 packets), we configure RAMs at 266MHz and TCAMs at 133MHz. Because a packet classification matching requires two TCAM

cycles as inputting a search key into TCAMs does, four TCAM chips are necessary and sufficient. Accordingly, a search key should be encoded within one RAM cycle to match the throughput of parallel TCAMs. To meet this requirement, either Approach 1 or 2 calls for six IFSMs, amounting to 246Kbit RAM for each; while the storage costs for FPES are given by TABLE II with totally 125Kbit RAM.

C. Distributed Policy Table Partition by Deriving-Merging

We incorporate the distributed storing strategy among TCAMs to minimize the storage cost. One crux is how to partition the entire policy table into sub groups. Notice that not any division is viable, unless it satisfies that: for any search key all the matched rules must be contained in one sub group. Besides, both the throughput and storage efficiency of the parallel distributed system depend highly on the extent to which the groups are partitioned evenly i.e. with uniform sizes (number of rules) and traffic load ratios. Most previous methods such as the "Prefix range-based", "Tree-based", and "Key-ID-based" ([6, 3, 16]) etc., merely aim at uniform group sizes, ignoring the distributed redundancy introduced by the overlapping among different sub groups as well as the traffic load balance which will directly impact the total throughput.

TABLE II. RAM REQUIREMENTS OF THE OPTIMAL SOLUTION (FPES)

	SIP& DIP	SP		DP		Protocol
		AR	EM	AR	EM	
Original speed (RAM cycle)	2	6	1	6	2	1
Parallel Num. of EUs	2	6	1	6	2	1
Parallel speed (RAM cycle)	1	1	1	1	1	1
RAM Costs (Block)	8× M4K	56× M512		128× M512		2× M512
Storage Req. (Kbit RAM)	32	28		64		1

We develop a novel method named "Deriving-Merging Partition (DMP)", which has the following virtues: 1) DMP is compatible with IPv6. It takes advantage of IPv6 features while is insensitive to the singularities of real-world policy tables; 2) DMP considers all the three objectives: the uniform group sizes, balanced load allocation and the minimum redundancy. DMP is executed in two steps: 1) the selection of Preliminary Partition Bits and 2) the Deriving-Merging Adjustment, as follows.

Selection of Preliminary Partition Bits:

We choose P bit-positions out of W-bit long rules as the Preliminary Partition Bits (PPB) to firstly partition the entire policy table into 2^P sub groups with possibly even sizes. Each sub group is identified by a '0-1' specification of the PPB. A brute-force approach is to traverse all the P-bit combinations out of W bits to get the optimal solution. However as W is 296 for IPv6, the computation for exhaustive search is too complex. We propose a set of efficient and universal methods to reduce both the computation complexity and the distributed storage:

Method 1: Since SP, DP contain ranges that need encoding and there is no prefix with the length between 64 and 128 bits (excluding 64 and 128 bits) ([17, 18, 19]), we eliminate these fields for further trials and concatenate all reserved bits to form a *Preliminary Selection Vector* (*PSV*).

Method 2: We denote by *TR* the total number of rules in the original policy table; by *PSV*[*i*] the *i*th bit-position in *PSV*; by *PSV*[*i*]_0, *PSV*[*i*]_1 and *PSV*[*i*]_* the numbers of rules whose *PSV*[*i*]*s* are '0', '1' or '*' respectively, then we have: *PSV*[*i*]_0 + *PSV*[*i*]_1 + *PSV*[*i*]_* = *TR*; and let *Dividing*[*i*]_0*or*1 be (*PSV*[*i*]_0*or*1 + *PSV*[*i*]_*) / *TR*. Due to distributed redundancy, the total number of rules in all the sub groups exceeds *TR*. So the ideal percentage of rules each sub group accounts for must be greater than $1/2^P$. Note that if *Dividing*[*i*]_0*or*1=*x%*, *Dividing*[*j*]_0*or*1=*y%*, the percentage of rules corresponding to (*Dividing*[*i*]_0*or*1 && *Dividing*[*j*]_0*or*1) is no more than *x%*×*y%*. Therefore we further remove from *PSV* the bit-positions whose *Dividing*[*i*]_0*or*1 are too trivial. For the sake of low distributed redundancy, we also discard the bit-positions whose *PSV*[*i*]_* are too significant. Now the *PSV* comprised of only the remaining bits is renamed "*Selection Vector* (*SV*)". To be more precise, a *PSV* bit remains in *SV* **iff**: *PSV*[*i*]_0 *or*1 > *MTh* **(2-1)** and *PSV*[*i*]_* < *RTh* **(2-2)**.

Both *MTh* and *RTh* are adjustable parameters. Our methods are pretty universal in that the values of *MTh* and *RTh* can be set elastically to achieve the tradeoff among the computation complexity, storage uniformity and distributed redundancy.

Deriving-Merging Adjustment (DMA):

DMA is invented to further optimize the results derived from PPB. For illustration, we suppose *SV* is 8-bit long and the 0th, 2nd and 4th bit-positions are PPB, then the resulting eight sub groups correspond to eight *Preliminary ID*s (*PID*): $0(000)_2$, $1(001)_2$, $2(010)_2$, $3(011)_2$, $4(100)_2$, $5(101)_2$, $6(110)_2$, $7(111)_2$. They finally evolve into *Derivative ID*s (*DID*) as shown in TABLE III, the '0's or '1's in *DID* are called "specified bits".

TABLE III. EVOLUTION FROM PID TO DID

PID	000	001	010	011
DID	0*0*0***	0*0*1***	0*1*0***	0*1*1***
PID	100	101	110	111
DID	1*0*0***	1*0*1***	1*1*0***	1*1*1***

PPB limits the distributed redundancy by *RTh*, but only the fixed P partition bits may be insufficient to ensure uniform group sizes and traffic load ratios. We expect the sub group sizes and traffic load ratios subject to the following two constraints[v]: let $GL[i]$ and $|GR[i]|_{(i=1,2,...N)}$ be the traffic load ratio and size of the sub group with *PID i* (*SG#i* for short), respectively; $Av(GL)$ and $Av(GR)$ denote the average traffic load ratio and the average size of all sub groups, respectively.

Constraint 1 (C1): $Max|GR[i]| / Av(GR) \leq RMA$

Constraint 2 (C2): $MaxGL[i] / Av(GL) \leq LMA$

Parameters *RMA* and *LMA* can be adaptively generated for different policy tables and the results of PPB. A reasonable and practical set of initial values is: *RMA* = 1.5, *LMA* = 2. And DMA is processed in two steps:

[v] As is analyzed later in this paper, the smaller the maximum size and traffic load ratio of all the resulting sub groups are, the less storage and better throughput our scheme produces.

1) Deriving: Sub groups that do not satisfy either of the two constrains are to be derived. We select one or two '*' bits from each *DID* of these groups as Additional Partition Bit(s) (APB) to divide each sub group into two or four *Derivative Groups*, each with smaller size and traffic load ratio than its ancestor.

2) Merging: After Deriving, the number of sub groups plus derivative groups will exceed 2^P. We select some intact sub groups with small sizes or traffic load ratios to merge with the derivative groups to form new sub groups. After Merging, the variances of both the sizes and load ratios of the 2^P sub groups are diminished. DMA is described below:

DMA Algorithm

i) **Find out** all sub groups that do not satisfy either **C1** or **C2** to be derived. Record them as: $SG\#a_1, SG\#a_2, \ldots, SG\#a_n$

ii) **for** *i* from 1 to *n*

{ **traverse '*'** in *DID* of $SG\#a_i$: one bit at a time

{ **if** ***DID***[*k*]='*'

and if *Dividing*[*k*]_0 *or* 1 satisfies **C1 and C2**

and if *Dividing*[*k*]_1 *or* 0 *can be merged under* **C1 and C2**

then Deriving-Merging at *DID*[*k*]; **return** }

traverse '*' in *DID* of $SG\#a_i$: two bits at a time

{ **if** the total size of certain derivative groups satisfies **C1 and C2**

and if the others can be merged without breaking **C1 and C2**

then { Derive-Merge; **return}** } }

iii) Record the *DID*s

Packet Identification:

An incoming packet must be identified to which sub group or TCAM it belongs. To guarantee an ultra high processing rate we need special identification scheme, since: 1) *DID*s may have different numbers of specified bits; 2) APB may come from different positions; 3) one sub group may correspond to more than one *DID*. To spare TCAM resources, we use RAMs for identification. A naïve method is to specify all '*'s in *DID*s by '0-1's and store all combinations. However this will incur enormous storage expansion. We design a hierarchy table scheme where PPB is first used to query the *Preliminary Table*, whose entry format is depicted in Fig.2.

TCAM ID	*BitPosition1*	*BitPosition2*

Figure 2. Entry Format of Preliminary Table

If *TCAM ID* ≠ 0, its value is just the final result indicating which TCAM the input belongs to; however if *TCAM ID* = 0 it implies that the PID is evolved and it is necessary to search for further information in the *Derivative Table*. The last two fields in the Preliminary Table indicating the positions of APB are used to address the Derivative Table where the final *TCAM ID* is stored. The entry format is shown in Fig.3.

TCAM ID

Figure 3. Entry Format of Derivative Table

The hierarchy table needs only 64 entries (16 for the Preliminary Table, 48 for the Derivative Table) in the worst case. The speed requirement of 4×OC768 is also easy to meet.

D. Global Redundancy-Based Parallel Adaptive Scheme

Four TCAMs working at 133MHz are adequate for 4×OC-768 line rate. For the sake of scalability and flexibility, we generally suppose *K* TCAMs are involved. The central problems are: 1) how to allocate the 2^P sub groups to the *K* TCAMs evenly; 2) how to dynamically balance the real-time traffic loads among TCAMs with relatively low storage costs. The solutions are presented in details below:

Distributed Allocation Scheme (DAS):

DAS distributes *N* sub groups to *K* TCAMs and strives to balance both storages and traffic loads among TCAMs. For clarity, we first describe the mathematical model of this multiple-objective problem. Let: $GL[i]$ and $|GR[i]|_{(i=1,2,\ldots N)}$ still be the traffic load ratio and size of $SG\#i$; $GR[i]$ be the set of rules in $SG\#i$; Q_k be the set of the sub groups assigned to TCAM $\#k$; $TL[k]$ and $TR[k]$ be the traffic load ratios and the number of rules of TCAM $\#k_{(k=1,2,\ldots,K)}$ respectively. Namely:

$$TL[k] := \sum GL[i], \quad TR[k] := |\bigcup GR[i]|, \quad (SG\#i \in Q_k)$$

The optimization problem is given by:

Minimize $Max(TL[i]\text{-}TL[j])$ & $Max(TR[m]\text{-}TR[n])_{(i,j,m,n=1,\ldots,K)}$

Note that the two objectives may contradict each other; this NP-hard problem can not be well solved by established algorithms. Assisted by the real-time balancing scheme discussed later in this paper, the traffic loads can be *Perfectly Balanced* if after DAS: $MaxTL[i] < 1/K_{(i=1,2,\ldots K)}$.

Then we elegantly convert the original multiple-objective problem into three single-objective ones at three levels: 1) optimize the storage objective with best efforts while perfectly balancing the traffic loads among TCAMs; 2) guarantee the perfect balance of traffic loads with higher priority while restricting the storage discrepancy; 3) balance the traffic loads with best efforts.

Level 1: At this level, we strive to optimize the two objectives simultaneously. We consider $TR[i]$ as the single objective and $TL[i]$ as an additional constraint. The mathematical model is:

Minimize: $Max(TR[i]\text{-}TR[j])_{(i,j=1,\ldots,K)}$

S.t.: $MaxTL[k] \le 1/K, Max(|Q_i| - |Q_j|) \le Min\{N\%K, 1\}$ [vi]

The algorithm is presented below:

Storage **E**fficiency **A**lgorithm (**SEA**)

STAGE i) **Sort** *N* sub groups in descending order of $GR[i]$s and record the result as $\{SG[1], \ldots, SG[N]\}$

STAGE ii) **for** *i* from 1 to *N* **do**

{ **Sort** $\{k, k=1,\ldots,K\}$ in ascending order of $TR[i]s$ and record as $\{TCAM[1], TCAM[2], \ldots, TCAM[K]\}$;

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

{ **if** $TL[TCAM[k]] + GL[SG[i]] \le 1/K$

and if $Max(|Q_i| - |Q_j|) \le Min\{N\%K, 1\}$

then ADD; **break; }}**

Level 2: In case SEA fails to find a feasible solution, we first ensure the perfect balance of traffic loads by $MaxTL[i] < 1/K$.

[vi] Notation '%' stands for the "mod" operation. if $N\%K=0$, $Min\{M\%K,1\} = 0$, which gives the minimum value of $Max\{|Q_i|-|Q_j|\}$; otherwise $Min\{M\%K,1\} = 1$ which is also the minimum value of $Max\{|Q_i|-|Q_j|\}$ in this case.

In addition, note that $|GR[i]|$s are quite close to one another after DMA, we introduce the constraint: $Max(|Q_i|-|Q_j|) \le VG$ to limit the discrepancy of $TR[i]s$. Parameter VG is initially set $Min\{N\%K, 1\}$ to target the optimal solution. When necessary, VG is increased iteratively until a feasible solution is found. Since the average of $|Q_k|$ is N/K, we suppose $\lfloor N/K \rfloor$ be the upper bound of VG. The mathematical model is:

Minimize: $MaxTL[i]_{(i=1,\ldots,K)}$

S.t.: $TL[k] < 1/K, Max(|Q_i| - |Q_j|) \le VG \quad _{(i,j,k=1,\ldots,K)}$

We sort the N sub groups in descending order of $GL[i]s$ and record the result as $\{SG[1],\ldots,SG[N]\}$. For clarity, we define the i^{th} *Allocation Unit* (AU#i) as the allocation of $SG[i+1]$, …, $SG[i+K]$, i=0,K,…. Since N may not be exactly divided by K, in this case a proper number of *Zero Groups* with $GL[i] = 0$ and $GR[i] = 0$ are supplemented to construct ($\lfloor N/K \rfloor$+1) AUs. In each AU, we sort TCAMs in ascending order of $TL[i]$s and record as $\{TCAM[1],\ldots,TCAM[K]\}$. In AU#$i$ $TCAM[1]$ should have got $SG[i+1]$ and $TCAM[K]$ got $SG[i+K]$ etc.; however when the variance of $TL[i]$s is too large, e.g., $TL[TCAM[K]] > TL[TCAM[1]]+GL[SG[i+1]]$, then function ***switch()*** assigns $SG[i+K]$ to $TCAM[1]$ instead of $TCAM[K]$ as an extra compensation. $TCAM[K]$ gets nothing in this AU to restrain the growth of $TL[TCAM[K]]$. Function ***supervisor()*** checks in advance whether the action of ***switch()*** will break VG constraint. If and only if the VG constraint will not be broken, it returns **PERMIT** and ***switch()*** acts. A more accurate description is given below:

Storage **C**onstrained **L**oad **E**fficiency **A**lgorithm (**SCLEA**):

STAGE i) **Sort** N sub groups in descending order of $GL[i]$s and record the result as $\{SG[1],\ldots,SG[N]\}$;

STAGE ii)

for i from 1 to $\lfloor N/K \rfloor$+1 **do**

{ **Sort** $\{k,k=1,\ldots,K\}$ in ascending order of $TL[k]$s and record the result as $\{TCAM[1],\ldots,TCAM[K]\}$;
Take $\{SG[(i-1)K+1],\ldots,SG[(i-1)K+K]\}$ to be allocated in the current AU; $pointer = (i-1)K+1$;

for k from 1 to K **do**

{ **if** $TL[K]$-$TL[k]$< $GL[SG[pointer]]$ **or** ***supervisor()*** **!=PERMIT**;
and if $TL[TCAM[k]]+GL[SG[pointer]] < 1/K$
then ADD; $pointer$++; **break;**
else if $TL[TCAM[k]]+GL[SG[pointer]] < 1/K$
then ***switch()***; $pointer$++; **break; } }**

Level 3: When neither SEA nor SCLEA gives a feasible solution indicating that the two objectives may be heavily mutual-exclusive, we only aim to minimizing $Max(TL[i]$-$TL[j])$ which is a more crucial concern. The algorithm is as follows:

Load **E**fficiency **A**lgorithm (**LEA**)

STAGE i) **Sort** N sub groups in descending order of $GL[i]s$ and record the result as $\{SG[1],\ldots,SG[N]\}$;

STAGE ii) **for** i from 1 to $\lfloor N/K \rfloor$+1 **do**

{ **Sort** $\{k,k=1,\ldots,K\}$ in ascending order of $TL[k]$s and record as $\{TCAM[1],\ldots,TCAM[K]\}$;
Take $\{SG[(i-1)K+1],\ldots,SG[(i-1)K+K]\}$ to be allocated in the current AU; $pointer = (i-1)K+1$;

for k from 1 to K **do**

{ **if** $TL[K]$-$TL[k]$<$GL[SG[pointer]]$
then ADD; $pointer$++; **break;**
else ***switch()***; $pointer$++; **break; } }**

The flow chart of DAS is depicted in Fig.4. According to the algorithms and flow chart, DAS can be summarized as follows: at the beginning SEA runs to aim for the optimal solution, i.e. to balance $TL[i]$s and $TR[i]$s simultaneously. If no feasible solution is returned, SCLEA operates next to ensure the perfect balance of traffic loads among TCAMs and restrict the discrepancy of $TR[i]$s. VG is loosened adaptively until a feasible solution is obtained; or if $VG \ge 3$ implying SCLEA is infeasible, LEA works to find a best-effort solution.

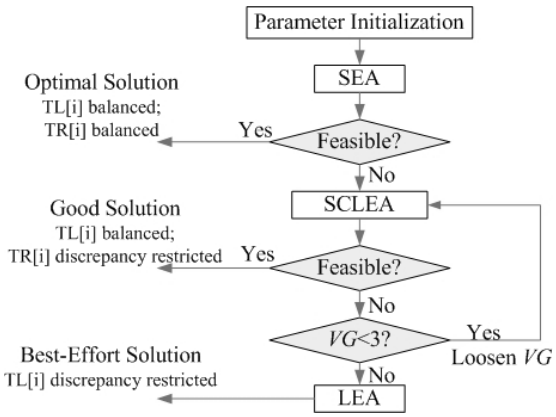


Figure 4. DAS Flow Chart

Real-time Load Balancing Scheme:

DAS is virtually a static solution in the sense that once the sub groups are pre-placed in different TCAMs the throughput will fluctuate along with traffic pattern changes. Our chief goal is an ultra high throughput; therefore we introduce the *Global Redundancy* (GR) to develop an efficient real-time balancing scheme as mentioned in Section II.A. In what follows we give the mathematical analysis of the balancing ability of GR.

Let $TL[k]$, $TG[k]$ and $D[k]$ $_{(k=1,2,\ldots,K)}$ stand for the Local Traffic ratio[vii], Global Traffic ratio and the overall load ratio assigned to TCAM#k respectively. Let A be the total amount of the Global Traffic load ratio. Then we have: $_{(i=1,\ldots,K)}$

$$TG[i] \ge 0, \ \sum_i TG[i] = A, \ \ D[i] = TL[i] + TG[i], \sum_i TL[i] + A = 1$$

The mathematical model of the adaptive balancing is given by:

To dynamically decide: $TG[i]_{(i=1,\ldots,K)}$
Minimize: $MaxD[i]$

[vii] Recall that "Global Traffic" refers to the traffic loads that correspond to GR; "Local Traffic" refers to the traffic loads that correspond to the sub groups allocated in DAS.

S. t.: $TG[i] \geq 0,\ \sum_i TG[i] = A,\ \sum_i TL[i] + A = 1$

Since $MaxD[i] \geq 1/K_{(i=1,\ldots,K)}$, the theoretically optimal solution is: $MaxD[i]=1/K \Rightarrow D[i]=1/K$. Then the mathematical model evolves into:

To dynamically decide: $TG[i]_{(i=1,\ldots,K)}$

Equalities:
$$\begin{cases} D[1]=TG[1]+TL[1]=1/K \\ \vdots \\ D[K]=TG[K]+TL[K]=1/K \end{cases}$$

S.t.: $TG[i] \geq 0_{(i=1,\ldots,K)}$

The above equalities have feasible solutions if and only if $TL[i]<1/K_{(i=1,\ldots,K)}$. And it is just the key condition under which the traffic loads can be perfectly adjusted thus resulting in a deterministic system throughput.

Selection of Global Redundancy:

It is obvious that the more sub groups we select as GR, the stronger balancing ability GR provides. However it is impractical due to the storage cost limitation. Define the *Traffic Density* of $SG\#i_{(i=1,\ldots,2^P)}$ as $\rho[i]=GL[i]/GR[i]$. The most efficient and economical way is to choose certain number of sub groups with the largest $\rho[i]$. Observing that the discrepancy of $GR[i]$s can be well bounded by DMP and is considerably minor compared to that of $GL[i]$s, we simplify $\rho[i]$ by $GL[i]$ and propose the first principle.

Selection Principle 1 (SP1): Sort the sub groups in decreasing order of $GL[i]$s and record the result as $\{SG[1],\ldots SG[2^P]\}$. We take $SG[1],\ldots,SG[R]$ $(1 \leq R \leq 2^P)$as GR, and label their traffic loads as $GRL[1],\ldots,GRL[R]$.

We expect the minimum value of R that ensures the perfect balance of traffic loads, i.e.: $TL[i]<1/K_{(i=1,\ldots,K)}$. According to DAS, the solution generated from SCLEA must satisfy $TL[i]<1/K$. We evaluate the worst-case performance of SCLEA without considering the constraint $TL[i]<1/K$; and then the minimum R is derived as follows:

Sort the N sub groups allotted in SCLEA in decreasing order of $GL[i]$s, and record the result as $\{GL[1],\ldots,GL[N]\}$ for simplicity. In AU#1, each TCAM deserves one sub group and $Max(TL[i]-TL[j])=GL[1]-GL[K]_{(i,j=1,\ldots,K)}$. Suppose when $i=\alpha$:

$$GL[\alpha+1]-GL[\alpha+K]=Max(GL[i+1]-GL[i+K])=\Delta_{\alpha,K}$$

. Due to the negative feedback characteristic of SCLEA, after AU $\#(\lfloor N/K \rfloor+1)$, the final maximum discrepancy Δ satisfies:

$$\Delta = Max(TL[i]-TL[j]) \leq \Delta_{\alpha,K} = GL[\alpha+1]-GL[\alpha+K] \leq GL[1]$$

When other $TL[i]$s are as small as possible except the largest one, it yields: $MaxTL[i] \leq \frac{1-A}{K}+\frac{K-1}{K}\Delta$ as the upper bound of $MaxTL[i]$. Thus we further have:

Selection Principle 2 (SP2):

$$MaxTL[i] \leq \frac{1-A}{K}+\frac{K-1}{K}\Delta \leq 1/K \Rightarrow A > (K-1)\times\Delta$$

Substituting A in **SP2** by $A=\sum_i GRL[i] \geq R \times GRL[R]$ yields: when $R>(K\text{-}1)\times GL[1]/GRL[R] \geq (K\text{-}1)$, SCLEA must have a feasible solution. In our 4×OC768-matching scheme K=4, thus R=3 is the *sufficient condition* to perfectly balance the traffic loads in any occasion. According to DMP, the resulting redundancy is at most 84%. In most occasions, R = 1 or 2 is a practical choice since the inequality has been loosened many times during the above mathematical deduction. We also prove later that under the assumptions in [16] the global redundancy is less than[viii] 56% in the worst-case.

IV. COMPLETE IMPLEMENTATION

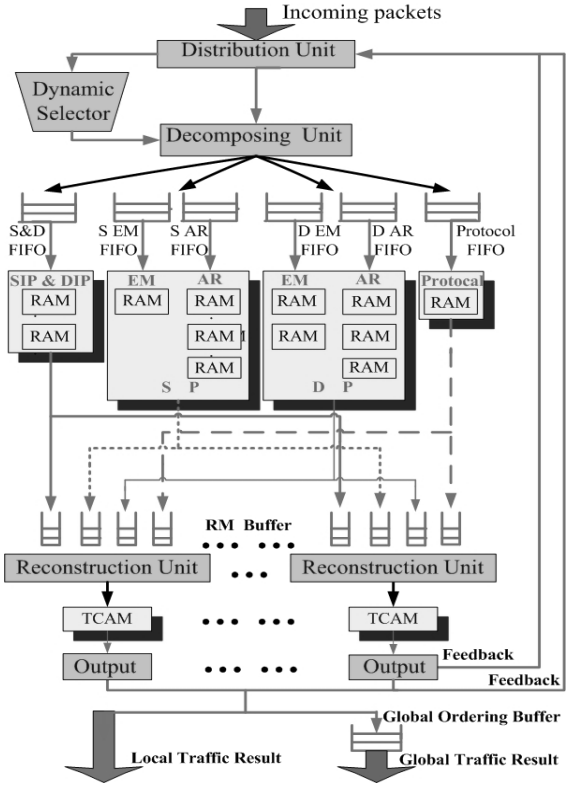


Figure 5. System Implementation

As mentioned in foregoing sections, we propose the packet classification scheme using the novel and efficient encoding algorithm FPES and partition approach DMP for IPv6-oriented 4×OC-768 line rate. Accordingly, we create the hierarchical table structure to identify the incoming packets. In addition, caching and labeling mechanisms are used to reduce delay and maintain the sequence order. Fig.5 shows the implementation in details and we will describe the main units in what follows.

Distribution Unit: It supports three major functions. 1) It identifies to which TCAM the incoming packet belongs by querying the hierarchy table. 2）It judges whether the packet is global traffic. If yes, the Dynamic Selector allocates this traffic to the least loaded TCAM and binds the packet and its dynamically allotted *TCAM ID* together to the Decomposing Unit; otherwise the packet is directly sent to the Decomposing Unit with its identified *TCAM ID* checked out from the hierarchy table. 3）It maintains one Serial Number (S/N) Counter for each TCAM. Counter[i] records the traffic loads

[viii] due to the overlapping between different sub groups

assigned to TCAM#i $_{(i=1,2,\ldots K)}$. Whenever a packet is allocated to TCAM #i, Counter[i] is incremented by one; and whenever a classification is over, Counter[i] is decremented by one based on the Output Unit feedback.

Decomposing Unit: This unit has three main tasks. 1) A DTag is generated to uniquely identify a packet and its associated TCAM. It is depicted in Fig.6. 2）It separates and encapsulates the five fields of a search key into the RAM Encoding Tags (RTag) as shown in Fig.7. 3） It pushes individual fields into the Search Key Encoding FIFOs in parallel. Recall that some fields may employ parallel EUs, this Unit schedules the traffic among the parallel EUs by simply using the *Round-Robin* method, as no distributed storing is adopted.

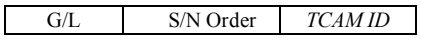
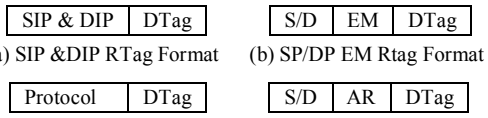


Figure 6. DTag Format



(a) SIP &DIP RTag Format (b) SP/DP EM Rtag Format

(c) Protocol RTag Format (d) SP/DP AR Rtag Format

Figure 7. RTag Format

S/D (one bit) is used to distinguish SP and DP considering that the code lengths of SP and DP are different

Search Key Encoding FIFO (SE FIFO): The SE FIFO is a group of FIFOs where the decomposed information (RTag) is held. Different RTag Formats correspond to different FIFOs: S&D FIFO, S/D_EM FIFO, S/D_ AR FIFO and Protocol FIFO. However, it should be noted that ideally no queueing is occurred under the configuration in this paper.

Range Matching Buffer (RM Buffer)&Reconstruction Unit: The major functions of this unit are as follows. 1）It receives encoded individual fields and reconstruct encoded keywords. Note that the reconstruction is performed in the RM Buffer immediately ahead of each TCAM. Therefore RAMs do not have to wait for each other during encoding. Each encoded field is sent to RM Buffer instantly after encoding. 2) RM Buffer is not a FIFO queue but a fast access register addressed according to Orders in DTags[ix], by which the slight disorder due to parallel processing is adjusted. Note that in ideal cases no disorder will occur. RM Buffer format is given in Fig.8.

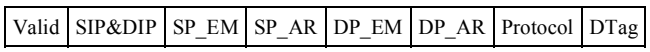


Figure 8. RM Buffer Format

The "Valid" bit is used to indicate whether this item has been reconstructed

Global Ordering Buffer: It is similar to RM Buffer and is used for adjusting the slight disorder of the Global Traffic.

V. PERFOMANCE EVALUATION

A. Throughput

In DAS, when either SEA or SCLEA derives the final allocation result, the TCAM traffic loads can be perfectly balanced by the GR scheme. When R = 3, SCLEA must have a feasible solution as analyzed before. In case R is too restricted to guarantee any feasible solution from SEA and SCLEA, we analyze the worst-case performance of LEA.

Sort the N sub groups allotted in LEA in decreasing order of $GL[i]$s. We suppose consecutive $GL[i]$s do not vary sharply under **Constraint 2** in DMA. More specifically, we give the following assumption and conclusion:

Assumption: The ***switch*()** function is not invoked more than once in one AU.

Conclusion: After LEA, we have:

$$\Delta = Max(TL[i] - TL[j]) \le GL[K \times (\lfloor N/K \rfloor - 1) + 1] \quad _{(i,j=1,2,\ldots K)}$$

Proof: We denote by Δ_i the $Max(TL[m]\text{-}TL[n])$ $_{(m,n=1,2,\ldots K)}$ after AU#i. At the beginning of AU#(i+1), there are two possible cases: 1) ***switch*()** does not act; 2) ***switch*()** acts. Case 1) indicates $\Delta_i \le GL[K \times i + 1]$ and after AU#(i+1), we have:

$$\Delta_{i+1} \le Max\{\Delta_i, GL[K \times i + 1] - GL[K \times i + K]\} \le GL[K \times i + 1] \quad .$$

Case 2) indicates that $\Delta_i \ge GL[K \times i + 1]$. Let $\Delta_i^{'}$ be the $Max(TL[m]\text{-}TL[n])$ $_{(m,n=1,2,\ldots K)}$ after the action of ***switch*()**. According to the **Assumption**, ***switch*()** will not act anymore in AU#(i+1). Similar to Case 1), we have:

$$\Delta_i^{'} \le GL[K \times i + 1] \Rightarrow \Delta_{i+1} \le Max\{\Delta_i^{'}, GL[K \times i + 1]\} \le GL[K \times i + 1]$$

In sum, in either case it holds that $\Delta_{i+1} \le GL[K \times i + 1]$. Notice that $GL[K \times i + 1]$ diminishes as i grows larger, LEA is proved well convergent.

If $N\%K < 2$, after LEA we have: $_{(i,j=1,2,\ldots K)}$

$$\Delta = Max(TL[i] - TL[j]) \le \Delta_{\lfloor N/K \rfloor} \le GL[K \times (\lfloor N/K \rfloor - 1) + 1]$$

If $N\%K \ge 2$, after LEA we have: $_{(i,j=1,2,\ldots K)}$

$$\Delta = Max(TL[i] - TL[j]) \le \Delta_{\lfloor N/K \rfloor + 1} \le GL[K \times \lfloor N/K \rfloor + 1] \le GL[K \times (\lfloor N/K \rfloor - 1) + 1] \quad .$$

According to **SP2** and $GRL[1] + \ldots + GRL[R] \ge R \times GRL[R]$, the sufficient condition for perfect load balance is:

$$R \ge (K-1) \times GL[K \times (\lfloor (2^P - R)/K \rfloor - 1) + 1] / GRL[R] \qquad (*)$$

Based on the assumption in [16]:

$$GL[x] \le GLR[R] \times (2^P - x)/2^P \ , 2K - R \le x \le N$$

When R = 2 or 3, $(*)$ holds, showing that 56% global redundancy (R=2) is sufficient in the worst-case.

B. Update

The updates in our scheme involve two independent parts that can be handled in parallel: the distributed policy tables in TCAMs and the encoding tables in RAMs. The former can be well dealt with by the existing approach called "CoPTUA"[15]. Our predefined bits-allocation scheme also efficiently supports incremental update for encoding tables in

[ix] Actually *relative orders* are used for reordering

RAMs. Since the updates for IP address and protocol are quite straightforward, we mainly discuss the update of range codes below.

For deleting a range code we first check whether there are no rules associated with it. If so, it is simply removed along with its code released (If this action results in an empty layer, that layer is removed）. For adding a new range, without the predefined bits-allocation manner such as the P^{2}C scheme [2], there are three cases: 1) When there is a *Free Layer* (where none of its existing ranges overlap with the new one) and there are free codes, we can directly allocate a free code to it. 2) When there are free layers but no free code remains on them, one of those layers must be selected and all the codes on it must be updated. 3) When no free layer exists, a new layer must be introduced. Under case 1) and 3), only the rules containing the new range need update. While in case 2), the conventional schemes will lead to the updates of all the existing range codes on the selected layer and hence a large number of rules are updated in TCAMs. As we use the predefined bits-allocation method, the update in case 2) becomes as simple as in other cases.

By virtue of the predefined bits-allocation method, adding new rules do not disturb the classification matching for other rules in TCAMs; only the newly added rules are concerned. What is more desirable, as we deploy FPES for each field, multiple EUs store the same encoding table for one field. When the encoding table in a certain EU is being updated, other EUs can maintain the encoding process incessant. Therefore this time consumption can be even ignored.

C. Packet Loss Probability

When the system is heavily loaded, packet loss may occur. One solution is to maintain RM Buffers with appropriate depth. The key concern is to find a relatively small depth value to make the loss possibility sufficiently minor. We suppose the K TCAMs and RM Buffers exhibit independent and identical long-run features in statistical sense then we can study a single pair of a TCAM and its RM Buffer. Since no packet disorder occurs in ideal cases, we can regard the RM Buffer as a FIFO queue. In light of the statistic fact that the packets arrive to a core router according to a Poisson Process with rate λ [20], we further suppose the traffic to a specific RM Buffer arrives according to a Poisson Process with rate λ/K when the traffic loads are distributed evenly among K TCAMs. Let D_T be the depth of each RM Buffer, $\{N_i\}_{i=1}^{\infty}$ be the stochastic process of the number of search keys in the buffer. We can model a TCAM and its RM Buffer as an $M/D/1/D_T$ queueing system with deterministic service rate μ . Let $\rho=\lambda/(K\times\mu)$ and $\alpha_j(\rho)=\sum_{i+m=j-2} e^{\rho(i+1)}(-\rho)^m(i+1)^m/m!(j\geq 2)$, then the Packet Loss Probability is: $P_L=P(N=n)=\frac{1+(\rho-1)\alpha_n(\rho)}{1+\rho\alpha_n(\rho)}$, $(n=D_T)$ [20].

Given the 4×OC-768 line rate and the configuration in this paper, it holds that $\mu \geq \lambda/K$. In the worst case when the traffic intensity is 100%, i.e. $\lambda = \mu$, the loss probability is well close to zero when $D_T \geq 5$.

D. Worst-case Processing Delay

We define the processing delay of an incoming packet as the time duration from the packet arrival to the instant when its classification is done. The worst-case processing delay is a vital metric since it impacts the system throughput, robustness and the packet loss probability. Thanks to FPES in RAMs and the deterministic performance of concerted TCAMs, the worst-case processing delay in our scheme is upper-bounded and reduced to the minimum as is analyzed below.

As in this paper the search key encoding in RAMs and classification matching in TCAMs are performed in series, the processing delay T_{pc} for the local traffic equals to the sum of the RAM encoding delay T_e and TCAM matching delay T_m. For each field, all parallel EUs store the complete encoding table thus eliminating queueing before the encoding module. Suppose the maximum encoding time of all the EUs is T_f, then $T_e=T_f$. Let D_T be the depth of each RM Buffer; T_c be the TCAM cycle. Then in the worst case $T_m= D_T\times 2T_c$ as each TCAM matching needs two TCAM cycles. Besides, $T_e \leq 2T_c$ in FPES, yielding $T_{pc}=T_e+T_m \leq 2T_c+ D_T\times 2T_c \leq (1+ D_T)\times 2T_c$.

For the global traffic, $T_{pc}=T_e+T_m+T_w$, where T_w stands for the waiting time in the Global Ordering Buffer. Assume the TCAM cycle is strictly constant and no latency exists within a single TCAM matching. Let N_i, N_{i+1} be the number of encoded keys in the RM Buffers before the i^{th} and $(i+1)^{\text{th}}$ arriving global traffic respectively. According to the GR-based real-time adjusting principle, it must hold: $N_i \leq N_{i+1}$ and $T_m^{\,i}= N_i\times 2T_c \leq T_m^{\,i+1}= N_{i+1}\times 2T_c$. Hence only when $N_i=N_{i+1}$, a trivial additional time T_w for reordering is incurred.

Set D_T =5 according to Section V.C and ignoring T_w, T_{pc} in our scheme is only $12T_c$ compared to $54T_c$ in [16].

E. Scalability and System Complexity

TCAMs provide deterministic lookup time independent of the rule length stored in a single TCAM slot. Only when a rule occupies M slots, does the lookup time increase by a factor M. For either more fields or less fields in a rule triggering the change of rule length, we can choose appropriate TCAMs with correspondingly longer or shorter slot length. By incorporating the range encoding, the system performance also remains stable regardless whether the fields in a rule are ranges, prefixes or exact numbers. It proves our system has a desirable scalability in various circumstances.

We achieve the unprecedentedly high throughput with only 49Kbit RAM and four TCAMs storing less than 1.56 times of a distributed policy table in practical cases; while under the same working frequency configuration of TCAMs, [16] may require eight TCAMs to match the IPv6 4×OC-768 line rate with extra storage to duplicate the encoding table in all TCAMs. Besides, the logics and buffers in our system are simple in that: 1) the flexible FPES enables identical encoding and TCAM matching throughput, nearly eliminating queueing before RAMs and TCAMs; 2) the efficient DAS and GR-based adjusting scheme balance the traffic among TCAMs, greatly shortening the RM Buffers; 3) the deterministic TCAM lookup time and real-time balancing principle make sequence disorder neglectable, minimizing the efforts for reordering; 4) the

search keys are identified by their *DID*s in a hierarchy table scheme which is easy to implement and costs little.

VI. EXPERIMENTAL RESULTS

In absence of IPv6 real-world data, we cannot experiment on IPv6 policy tables directly. However, as the key algorithms in our scheme, say DMP and DAS, are independent of and insensitive to the patterns of policy tables, we use IPv4 rule database set#5[13] to inspect the efficiency of DMA and DAS.

A. Selection of Preliminary Partition Bits

For IPv4, *PSV* is originally 72-bit long. We set *MTh*>500 and *RTh*<150, the length of *SV* decreases significantly to 15-bit after **Method 2**. The resulting sizes of sub groups are shown in Fig.9. The maximum *GR*[*i*] is 205 and the total number of rules is 2037. Both are much better than those in [16].

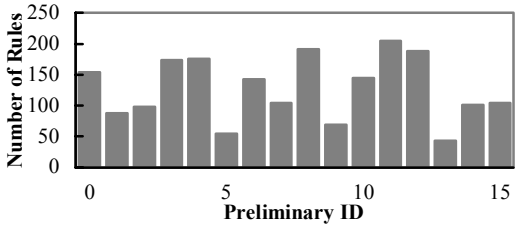


Figure 9. Sizes of Sub Groups after PPB

B. Deriving-Merging Adjustment

We further optimize the partition by DMA with *RMA*=1.5, *LMA*=2. For better proofs of the system throughput and robustness, we will generate *GL*[*i*]s later with huge unbalance, so we only consider *RMA* here without balancing *GL*[*i*]s. The result is in Fig.10, with totally 2143 rules, *MaxGL*[*i*]=147 and *MinGL*[*i*]=104. Detailed information is given in TABLE IV.

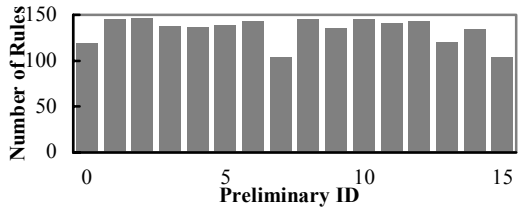


Figure 10. Sizes of Sub Groups after DMA

C. Distributed Allocation Scheme

We select the traffic load pattern with large variance as shown in Fig.11 to create a difficult case for load-balancing.

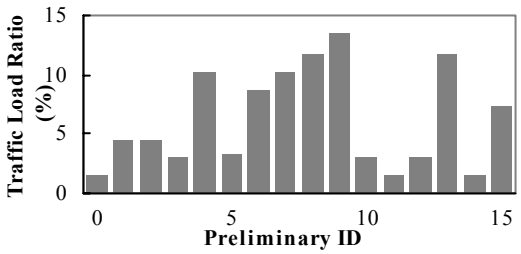


Figure 11. Traffic Load Pattern

GL[8] and *GL*[9] are chosen as GR. SCLEA derives the final result as shown in TABLE V.

TABLE IV. DETAILED INFORMATION ON DMA

Dividing Group ID	**0**	**3**	**4**
Derivative ID	0000&0(SV[69])	0011&0(SV[56])	0100&1(SV[33])
Original Size	154	173	176
Deriving Size	119	138	137
Merging Group ID	**14**	**2**	**1**
Derivative ID	1110; 0000&1(SV[69])	0010; 0011&1(SV[56])	0001; 0100&0(SV[33])
Original Size	101	98	88
Merging Size	135	147	146
Dividing Group ID	**8**	**11**	**12**
Derivative ID	1000&0(SV[32])	1011&1(SV[60])	1100&1(SV[47])
Original Size	192	205	188
Deriving Size	145	142	143
Merging Group ID	**9**	**5**	**13**
Derivative ID	1001; 1000&1(SV[32])	0101; 1011&0(SV[60])	1101; 1100&0(SV[47])
Original Size	69	54	43
Merging Size	136	139	120

TABLE V. RESULTS OF DAS

TCAM	**Sub Group ID (Excluding GR)**				**Num. of Rules**	**Traffic Load Ratio**
#1	13	5	10		306	18.13%
#2	4	2	12		397	17.9%
#3	7	1	3	11	474	19.32%
#4	6	15	14	0	464	19.36%

According to the result, *MaxGL*[*i*]<1/*K*=25%, the classification loads can be perfectly adjusted by the adaptive balancing scheme. The number of rules after DAS is 1641; the global redundancy is $4\times(GR[8]+GR[9])=1124$ rules. So the total number of rules is 2765, indicating 78% redundancy in all including the distributed redundancy; while the scheme in [16] incurs more than 80% redundancy but achieving only a quarter of our throughput for IPv4 packet classification.

D. Supplemental Results and Discussions

We derive the experimental results of the largest IPv4 rule database set#5 above, demonstrating that for the policy tables with large numbers of rules, DMP exhibits obvious efficiency and advantages over their counterparts in similar researches. For supplemental experiments, other rule sets may contain inadequate rules (far less than the number of rules in set#5) so that the distributed storing strategy losses its effectiveness; and

a low variance will trigger distinct unbalance among the small sub group sizes. However, in what follows the small rule sets are useful to reveal the efficiency of DMA from other perspectives. We take the rule set#4 with 264 rules for further illustration by comparing the results with and without DMA.

Without DMA, PPB derives the final result hence both the balance among sub group sizes and distributed redundancy are completely subject to the selection of PPB. However in many occasions the two aspects mutually conflict. Table VI gives the results of dividing rule set#4 by PPB only, where "Case1, 2, 3" represent the results of minimum variance of sub group sizes, minimum distributed redundancy and a tradeoff between "Case1, 2" respectively. The extremes of "Case1 and 2" in Table VI prove the confliction mentioned above and the inefficiency of dividing by PPB only.

TABLE VI. DIVIDING RESULTS ONLY BY PPB

	Rule Set#4 (264 rules)						
PID	**Case1 (rules)**	**Case2 (rules)**	**Case3 (rules)**	**PID**	**Case1 (rules)**	**Case2 (rules)**	**Case3 (rules)**
0	59	57	23	**8**	52	0	23
1	60	1	2	**9**	46	0	2
2	52	1	36	**10**	54	0	31
3	44	1	72	**11**	54	0	68
4	55	1	23	**12**	68	0	52
5	38	1	2	**13**	52	0	2
6	52	37	32	**14**	68	0	32
7	37	165	36	**15**	61	0	74
Sum	Case1: 852 rules; Case2: 264 rules; Case3: 510 rules						

In contrast, with the further balancing of DMA we can solely pursue the minimum redundancy when choosing PPB. Therefore we choose Case 2 in Table VI due to its minimum redundancy, and *SG*#0 is derived by one deriving bit while #6 and #7 by two bits for each. The final sub group sizes after DMP are given by: {17, 23, 15, 22, 25, 30, 91, 101, 1, 1, 1, 1, 1} with overall 324 rules; or we derive all the three sub groups *SG*#0, *SG*#6 and *SG*#7 by two bits for each, and the sub group sizes after DMA are: {17, 23, 15, 22, 18, 23, 24, 26, 63, 62, 67, 68, 1, 1, 1, 1} with overall 428 rules.

So far it is clear that DMA not only optimize the balance of sub group sizes and the distributed redundancy, but also achieve a better tradeoff between the two metrics when they severely conflict with each other. The flexibility of DMA also comes from the variable numbers of deriving bits we use: the distributed redundancy is smaller when using fewer deriving bits while the sub group sizes become more uniform when using more deriving bits. In fact, we even do not restrict the number of sub groups to be 2^P in pursuit of a better flexibility.

VII. CONCLUSION

Facing the increasing line rates and the coming era of IPv6, an even higher throughput matching $4\times$OC-768 line rate and IPv6-oriented packet classification are arising as two principle focuses. In this paper we address both the problems. Our scheme employs FPES to maximize TCAM utilization and throughput. Besides, DMP, independent of the particularities of real policy tables or traffic patterns, is originally proposed to perfectly partition the IPv6 policy tables. We further introduce delicate GR approach to adaptively adjust the traffic allocation among TCAMs. Theoretical analyses show that the deterministic system throughput can be well guaranteed and the worst-case throughput is upper-bounded with relatively low redundancy. The worst-case delay, loss probability and the update rate are also superior compared to similar researches. The experimental results further demonstrate the great efficiency and advantage of our novel algorithms.

REFERENCE

[1] M. E. Kounavis, A. Kumar, H. Vin, R. Yavatkar, and A. T. Campbell, "Directions in Packet Classification for Network Processors," in Second Workshop on Network Processors(NP2),February 2003

[2] J. van Lunteren and A. P. J. Engbersen, "Fast and scalable packet classification". IBM Res. Rep., RZ 3210, 2000

[3] Thomas Y.C.Woo, "A Modular Approach to Packet Classification: Algorithms and Results". *Proc. of IEEE INFOCOM*, 2000.

[4] IDT,"IDT75P52100 Network Search Engine". June 2003

[5] CYPRESS, "CYNSE10512 Network Search Engine". November 2002

[6] Rina Panigrahy, Samar Sharma, "Reducing TCAM Power Consumption and Increasing Throughput", *Proc of IEEE Hotl'02*

[7] David Edward Taylor, "Models, algorithms, and architecture for scalable packet classification". Phd Thesis, Saint Louis, August 2004

[8] "CERNET BGP View: 6Bone BGP Routing Table Statistics". 2005

[9] http://www.iana.org/assignments/ipv6-unicast-address-assignments, Feb 2005

[10] David E. Taylor, Jonathan S. Turner, "ClassBench: A Packet Classification Benchmark". WUCSE-2004-28, May 2004.

[11] http://www.arl.wustl.edu/~det3/ClassBench/.

[12] Guansong Zhang et al., "Fast Packet Classification Using Field-Level Trie". Globecom '03, 2003

[13] The authors would like to thank Professor Jonathan Turner and Mr. David Taylor from Washington University in St. Louis for kindly sharing their real-world databases

[14] 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.

[15] 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

[16] K. Zheng, H. Che, Z. Wang and B. Liu, "TCAM-based Distributed Parallel Packet Classification Algorithm with Range Matching Solution", *IEEE INFOCOM* , March, Miami, 2005.

[17] RFC 3578: R. Hinden, S. Deering and E. Nordmark, "IPv6 Global Unicast Address Format", August 2003, available at: ftp;//ftp.ripe.net/rfc/rfc3578.txt

[18] RFC 267: APNIC, ARIN, RIPE NCC. IPv6 Address Allocation and Assignment Policy, Document ID: ripe-267, January 2003, avaible at: http://www.ripe.net/rip/docs/ipv6policy.html

[19] RFC 3177: IAB, IESG, "IAB/IESG Recommendations on IPv6 Address". September 2001, available at: ftp://ftp.ripe.net/rfc/rfc3177.txt

[20] S. Alouf, P. Nain, D. Towsley, "Inferring Network Characteristics via Moment-Based Estimators", available at: http://www-sop.inria.fr/maestro/personnel/Sara.Alouf/Publications/infer.pdf