# A TCAM-Based Distributed Parallel IP Lookup Scheme and Performance Analysis

Kai Zheng, *Student Member, IEEE*, Chengchen Hu, *Student Member, IEEE*, Hongbin Lu, *Student Member, IEEE*, and Bin Liu, *Member, IEEE*

***Abstract*—Using ternary content addressable memory (TCAM) for high-speed IP address lookup has been gaining popularity due to its deterministic high performance. However, restricted by the slow improvement of memory accessing speed, the route lookup engines for next-generation terabit routers demand exploiting parallelism among multiple TCAM chips. Traditional parallel methods always incur excessive redundancy and high power consumption. We propose in this paper an original TCAM-based IP lookup scheme that achieves both ultra-high lookup throughput and optimal utilization of the memory while being power-efficient. In our multi-chip scheme, we devise a load-balanced TCAM table construction algorithm together with an adaptive load balancing mechanism. The power efficiency is well controlled by decreasing the number of TCAM entries triggered in each lookup operation. Using four 133 MHz TCAM chips and given 25% more TCAM entries than the original route table, the proposed scheme achieves a lookup throughput of up to 533 MPPS while remains simple for ASIC implementation.**

***Index Terms*—IP, power consumption, route lookup, TCAM, throughput.**

## I. Introduction

THE challenge of IP address lookup arises from the facts that 1) the length of IP prefix is variable; given that an IP address may match multiple prefixes in the table, *Classless Inter-Domain Routing* (CIDR) dictates the *Longest Matching Prefix* (LMP) should be chosen; and 2) advances in fiber-optic technology are pushing the line rate of core routers to 40 Gbps or even higher, which implies that a single line card would support the packet processing rate of 128 MPPS (million packets per second) or higher. Moreover, in some cases, a lookup mechanism may be designed to serve multiple line cards (for example, the Cisco 10000 series routers), which poses further stress on the lookup components.

Currently, packet forwarding based on CIDR IP addresses is well understood with both trie-based algorithms and TCAM-based schemes. The original Trie structure [1] used to solve the LMP problem is a time-consuming algorithm. The PATRICIA-Trie algorithm [2] widely used in the BSD kernel reduces the average memory accesses of the original Trie by introducing the so-called path-compressed trie structure; however, it still requires 32 memory accesses for an IP address lookup in the worst case. S. Nilsson *et al.* proposed the Level Compressed Trie algorithm [3], in which they used hash table to replace the full sub-tries, and the expected number of memory accesses was further reduced, but it still needed 32 memory accesses in the worst case. The Multi-bit Trie algorithms inspect several bits of the key in one memory access, and so the lookup time can be significantly reduced; however, many redundant nodes are added, and the space requirement is very large. Gupta *et al.* proposed a hardware-based multibit tries lookup scheme called DIR-21-3-8 [4]. In this scheme, only three memory accesses are needed in the worst case, but more than 9 MB memory space is required, so only DRAM with large capacity can be adopted. The Lulea algorithm [5] proposed by M. Degermark *et al.* and the BC-16-16 scheme [6] proposed by N. Huang *et al.* both used a so-called Bitmap Compression Technique to optimize the multibit trie algorithm. The idea is that the bitmap structures can be introduced to represent part of the trie and the memory requirement can be significantly reduced. Hence, the forwarding table can be small enough to fit into the CPU cache line or fast SRAM, and the lookup speed can be increased; however, both of them are not scalable and have very high update complexity, due to the requirement of leaf-pushing. W. Eatherton's tree-bitmap algorithm [7] improves upon Lulea by creating a data structure (with two kinds of bitmaps, internal bitmap and external bitmap) which does not require leaf-pushing and can be traversed with a single wide memory access per node while supporting fast incremental updates. An implementation of the Tree Bitmap algorithm, referred to as *Fast IP Lookup* (FIPL) [8], shows that a storage requirement of 6.3 bytes per prefix and a performance of over one million lookups per FIPL engine can be achieved, meanwhile the design also supports up to eight parallel engines, capable of scaling the throughput to nearly 7–8 million lookups per second. However, because of their intrinsic characteristics, the lookup speed in trie-based mechanisms using DRAMs or SRAMs can hardly be increased further.

*Ternary content addressable memory* (TCAM) is a fully associative memory that allows a "don't care" state to be stored in each memory cell in addition to 0's and 1's. It is a promising device to build a high-speed LMP lookup engine, because it returns the matching result within a single memory access time. Moreover, maintaining the forwarding table in TCAM-based schemes is generally simpler than that in trie-based algorithms. However, the high cost-to-density ratio and low power efficiency of the TCAM are traditionally the major concerns in building the lookup engine. Recent developments in

Manuscript received June 21, 2004; revised November 7, 2004, and April 27, 2005; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Buddhikot. This work was supported by NSFC (No.60173009 and No.60373007) and National 863 High-tech Plan (No. 2002AA103011-1). A preliminary version of this work was presented at the 23rd IEEE INFOCOM, March 2004.

The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: zk01@mails.tsinghua.edu.cn; hucc03@mails.tsinghua.edu.cn; lu-hb02@mails.tsinghua.edu.cn; liub@tsinghua.edu.cn.)

Digital Object Identifier 10.1109/TNET.2006.880171

TCAM technology have effectively addressed the issue of high cost-to-density ratio. There are commercially available TCAM devices with density up to 18 Mbits per chip and the costs are now comparable with the trie-based schemes [9]–[11]. However, on the other hand, although the TCAM can finish each lookup within a single memory access, the lookup throughput is still restricted by the TCAM accessing speed. Though current TCAM can work at a speed up to 266 MSPS[1] [9], it still cannot guarantee sufficient TCAM accessing bandwidth, or throughput, for next-generation terabit routers. Because, firstly, in many cases the TCAM-based lookup mechanism should also support other functions in addition to IP lookup at the same time, e.g., packet classification, etc., so the whole TCAM accessing bandwidth may not be totally available for IP route lookup. Second, in some practical cases, multiple line-cards may share the same TCAM-based lookup mechanism to save system cost. Finally, when scaled to support IPv6, multiple TCAM accesses would be needed for a single address lookup because of the distinctly increased search key length. All of these result in the demand for more powerful lookup engines with scalable throughput.

Many studies have also been conducted to optimize the TCAM-based lookup engines [12]–[16]. Liu *et al.* [12] used both pruning techniques and logic minimization algorithms to reduce the size of TCAM-based forwarding tables, which in turn reduces the cost and power consumption of the TCAMs. However, the power consumption is still quite high. Zane *et al.* [13] developed a power-efficient lookup engine benefited from a new feature of some TCAMs called "partition-disable". The idea is that during a parallel lookup operation, not all entries in the table are compared. If a large forwarding table can be partitioned into small partitions and only the one containing the prefixes that match the incoming IP address is enabled during the lookup operation, the power consumption of the TCAM can be dramatically reduced. However, it requires the hardware support of the new feature.

The most important index of a high-performance lookup engine is its lookup throughput, in our point of view. However, almost all of the algorithms mentioned above only strived to reduce the latency of each lookup operation, including conventional TCAM solutions, in which Entry-Level Parallelism (ELP) is adopted. As mentioned before, it still has a chance that the currently "high-enough" lookup throughput may not be sufficient for future high-end applications. Without developing Chip-Level Parallelism (CLP), the lookup latency can at most be reduced to one memory access time; therefore in these cases, because of the reciprocal relationship between lookup throughput and latency, memory access time will dominate the lookup performance (throughput) in this kind of schemes. As a matter of fact, an observation shows that the memory access time is reducing about 7% per year [21], which can hardly catch up with the increasing rate of the wire-speed of the optical interfaces, roughly doubling every year [21]. Thus, developing CLP to provide scalable throughput seems to be a preferable solution to break the restriction of memory access time and achieve high lookup throughput.

We presented a simple trie-based distributed parallel lookup algorithm in [17]. Using four to eight memory chips for parallel lookup operations and assuming that the traffic is evenly distributed among IP prefixes, the proposed scheme achieves a lookup speedup factor of 3 to 7, while the memory requirement remains unchanged. Nevertheless, due to the facts that the traffic load among the IP prefixes is not so evenly distributed, the lookup throughput becomes non-deterministic. Panigrahy *et al.* proposed a multi-chip mechanism to improve lookup throughput [15]. In their scheme, eight or more TCAM chips are used for parallel lookup operation. Based on certain assumptions of the traffic distribution and by doubling the number of entries (duplicating the TCAM contents), a speedup factor of 5 is achieved. However, eight chips need eight separate data buses (a large pin count) and the traffic distribution ASIC needs to work at a frequency of $8*133$ MHz $>1$ GHz, which is difficult to implement. The doubled number of route entries also implies the increase of the cost and power consumption, which makes it non-scalable.

In this paper, we propose an ultra-high-throughput and power-efficient IP lookup scheme using commercially available TCAMs to satisfy the demand of next-generation terabit routers. Our scheme employs distributed organized multiple memory modules for parallel table lookup operations, in order to break the restriction of the TCAM access speed and to reduce the power consumption. We have also devised the algorithms to solve the two main issues in applying chip-level parallelism. The first issue is how to evenly allocate the route entries among the TCAM chips to ensure a high utilization of the memory. The second one is how to balance the lookup traffic load among the TCAMs to maximize the lookup throughput.

The rest of this paper is organized as follows. Section II presents a distributed TCAM table organization and the load-balance-based table construction algorithm. Section III describes the complete architecture of the proposed ultra-high-throughput and power-efficient IP lookup engine. Section IV presents the theoretical performance estimation based on queueing theory as well as the simulation results of the proposed scheme. Section V gives the concluding remarks.

## II. Distributed TCAM Organization and Load-Balance Table Construction Algorithm

Our scheme exploits the parallelism among multiple TCAMs to increase the lookup throughput. By analysis of the existing route tables, we first introduce a method to divide the forwarding table into small segments. A load-balance-based table construction algorithm is then proposed to evenly allocate these segments into TCAMs while keeping the lookup traffic load balanced among them. At the end of this section, we introduce the concept of *dynamic power efficiency* for the power consumption analysis.

### A. Terminology and Definition

*Definition 1:* The bits in an IP address are ordered as shown in Fig. 1, where the 1st bit (MSB) lies in the leftmost position and the 32nd bit (LSB) lies in the rightmost position.

[1]SPS: search per second. Here, actually, the working frequency of the TCAM chip is 133 MHz, and the accessing throughput for the search key is 133 Mkeys per second. The index 266 MSPS comes from the fact that by certain new features of TCAM products, two parallel searches in different databases can be performed simultaneously for a given input key.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | ...... | 31 | 32 |

Fig. 1. IP address format.

*Definition 2:* An *extended forwarding table* is derived from the original forwarding table by expanding each prefix of length $L$ ($L < 13$) into $2^{13-L}$ prefixes of length 13.[2] For example, prefix $\langle 10.0.0.0/8 \rangle$ is expanded into $2^{13-8} = 32$ prefixes as below:

$\langle 10.0.0.0/13 \rangle$ 00001010.00000***.********.********;
$\langle 10.8.0.0/13 \rangle$ 00001010.00001***.********.********;
......
$\langle 10.248.0.0/13 \rangle$ 00001010.11111***.********.********.

It is straightforward that the extended forwarding table is equivalent to the original one. Since the number of prefixes shorter than 13 bits is quite small (which is why we adopt 13 as the threshold here), the expanding operation introduces few additional route entries.[3] We introduce this operation to facilitate the forwarding table segmentation as described in Definition 4. Without special notice, the *forwarding table* in the rest of the paper refers to the extended forwarding table.

*Definition 3:* The proposed scheme may contain redundant entries. Let $N_0$ be the number of the entries in the original forwarding table, $N$ be the number of the entries in the extended table ($N_0 \approx N$), and $M$ be the actual number of entries in the table of the proposed scheme. $Prefix[i]$ denotes the $i$th prefix in the extended table, where $i$ is from 1 to $N$.

*Definition 4:* A concept of ID (segment) is defined to classify the prefixes in the forwarding table. We define a bit-string extracted from the first 13 bits of a prefix as its ID. For instance, if we define the 10–13th bits of a prefix its ID, then the ID of prefix $\langle 10.16.0.0/13 \rangle$ is "0010" (2) and the ID of prefix $\langle 10.36.5.0/24 \rangle$ is "0100" (4). We say $Prefix[i] \in j$, when the ID of $Prefix[i]$ equals $j$. We also use the ID segment to identify the ID group, the set of the prefixes with the same ID.

*Definition 5:* Because the lookup traffic distribution among the IP prefixes can be derived by certain statistical method, we assume that it is known here. $D\_prefix[i]$ is defined as the ratio of the traffic load of $prefix[i]$ to the total bandwidth. Therefore

$$\sum_i D\_prefix[i] = 1. \tag{2.1}$$

*Definition 6:* We introduce a concept of storing *redundancy rate* here. The redundancy rate of a specific prefix (or ID group) is defined as the number of duplications of this prefix (or ID group) stored in the TCAM chips. The redundancy rate of the entire scheme is $M/N$.

*Theorem 1:* In the (extended) forwarding table, two prefixes with different IDs do not need to be stored in any specific order when applying the longest prefix matching operation. As long as the prefixes in an ID group are stored in the decreasing order of prefix lengths, the TCAM(s) will return the correct matching result.

*Proof:* TCAM-based matching requires that Prefix 1 should be stored in lower address than Prefix 2 if Prefix 2 is the prefix of Prefix 1; but two prefixes each one of which is not a prefix of the other can be stored in arbitrary order. Since each prefix in the extended forwarding table has a length of 13 or more, the first 13 bits of two prefixes with different IDs must not be a prefix of each other. Therefore, they need not to be stored in any specific order. □

## B. Analysis of the Real-World Route Table

We have analyzed the route table snapshot data provided by the IPMA[4] project and found out that the route prefixes can be evenly split into groups according to their IDs, so long as we make an appropriate ID-bit selection. This approach will contribute to balancing the storage among the TCAM chips and result in high-capacity utilizations of them (since identical-sized chips are usually adopted in most practical cases).

A brute-force approach to find the right set of ID bits would be to traverse all of the bit combination out of the first 13 bits of the prefixes, and then measure the splitting results to find out the most even one. However, this may be quite an expensive computation, since the number of prefixes may be fairly large.

According to our analysis and experiment results, three heuristic rules can be adopted to significantly reduce the traversing complexity. 1) The width of (or, the number of bits selected as) ID need not to be large, e.g., four to five bits are quite enough for most cases. 2) The number of patterns formed by the first 6 bits of the prefixes is quite unevenly distributed according to our experiments with real-world route tables. For the sake of finding an even classification of the prefixes, it is best not to adopt these (i.e., the 1st–6th) bits. 3) If the traverse of the combination of successive bits can find reasonable even results, there is actually no need to further traverse other non-successive bit combinations.

Based on these three heuristic rules, we easily found out reasonable ID-bit selecting solutions for four well-known real-world prefix databases, all of which adopt the 10th–13th bits, and the prefixes are classified into $2^4 = 16$ groups, respectively. The results are depicted in Fig. 2. All of the four tables show quite even distribution.

Notice that the ID-bit selecting scheme is totally flexible. Both the number of ID bits and their positions can be freely chosen according to the practical route tables. With the three heuristic rules, we produce satisfactory even results in almost all real-world cases collected by IPMA. And according to our experimental results, it tends to be that the larger the route table is (e.g., the BGP route tables of core level routers), the better the result we can achieve.

## C. Distributed Memory (TCAM) Organization

We have introduced a simple but efficient method to partition the forwarding table into small segments by using the ID bits of the prefixes. Since multiple TCAMs are used in our scheme, the next step is to allocate these ID segments (groups) to each TCAM. Based on the analysis of the route table above, we know that the prefixes are evenly distributed into each ID group, the

[2]Use the rule of LMP when the expanded prefixes overlapped.

[3]We extend the routing table of Mae-West router and get no more than 2% additional entries.

[4]A joint effort of the University of Michigan and Merit Network, available at http://www.merit.edu/ipma

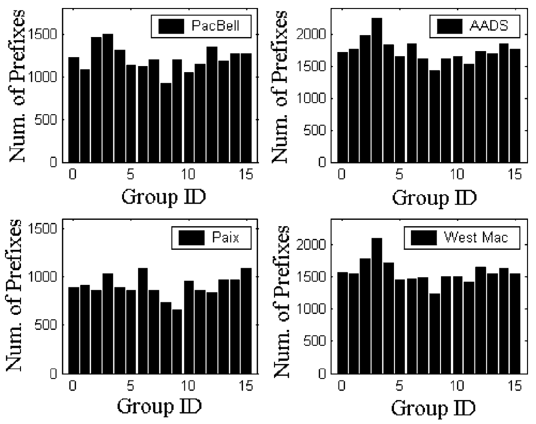


Fig. 2. Prefix distribution among ID groups of four real-world route tables.

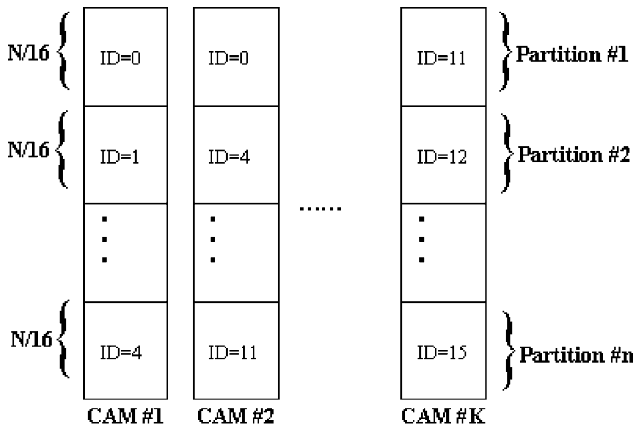


Fig. 3. An example of the TCAM organization.

size of which is approximately $N/16$. Therefore, we may partition each TCAM into small blocks with a size of $N/16$ (or slightly larger) so that each block can store one ID group. Now we use the ID group as the basic unit to allocate the prefixes among the TCAMs. As Theorem 1 dictates, as long as the prefixes in an ID group are arranged in decreasing order of prefix length, the TCAM will return the LMP as the result. Fig. 3 shows an example of the TCAM organization.

## D. Load-Balance-Based Table Construction (LBBTC) Algorithm

Allocating the prefixes evenly and balancing the lookup traffic among the TCAMs are the two tasks of the proposed table construction algorithm. We have addressed the first task in the subsection above. For the second task, we first calculate the load distribution of the ID groups, $D\_id[j]$, $(j = 1, \ldots, 16)$, by summing up the distribution of the prefixes in the same ID group:

$$D\_id[j] := \sum_{\text{prefix}[k] \in j} D\_prefix[k]. \tag{2.2}$$

Two methods can be then introduced to balance the lookup traffic among the TCAMs. First, we can use $D\_id[j]$, $(j = 1, \ldots, 16)$ as the weights of the ID groups and make the sum of the weights of the ID groups in each TCAM balanced. Second, for the ID groups with large weights, we may introduce storing redundancy into the scheme. By duplicating the prefixes in a specific ID group to multiple TCAMs, we distribute the traffic of the ID group among these TCAMs. For example, if ID group $j$ is allocated (duplicated) to three TCAMs, then each one of these three chips bears a traffic load of $D\_id[\text{j}]/3$. Our proposed algorithm is the combination of these two methods. Before we describe the table construction algorithm, a mathematical model of the problem is introduced.

*1) Mathematical Model of the Problem:* Let $K$ be the number of TCAM chips; $Rd$ be the storing redundancy rate of the whole mechanism, $T$ be tolerance of the discrepancy of table sizes among the TCAM chips, and $P$ be the length of the ID segment (it is assumed to be 4 in the proposed mechanism), and there should be $2^P$ ID groups. $K$, $Rd$, $T$, and $P$ are given.

Let $S$ be the set of all the ID groups, $S = \{1, 2, \ldots, 2^P\}$, $Q_k$ $(k = 1, \ldots, K)$ be the set of the ID groups that are allocated to TCAM chip $\#k$, $\forall Q_k$, $Q_k \subseteq S$ and $\cup Q_k = S$ $(k = 1, \ldots, K)$, and $|Q_k|$ denotes the number of elements (ID groups) that $Q_k$ contains.

We define the number of TCAMs that ID Group $\#j$ is allocated into as $G[j]$ $(j \in S)$, and $G[j]$ also denotes the storing redundancy rate of the ID group $\#j$. We define $W[j]$ as the incremental traffic load distributed to a TCAM chip when ID group $j$ is allocated to this chip, $W[j] := D\_id[j]/G[j]$, $(j \in S)$, and we call $W[j]$ the partition-load of ID group $\#j$.

We define $D[k]$ as the traffic load allocated to TCAM chip #k, $D[k] := \sum_{j \in Qk} W[j]$ $(k = 1, \ldots, K)$; then the optimization problem is given by:

**The Load-Balance-Based Table Construction Problem**

***Subject To:***

$$Q_j \subseteq S,\ j = 1, 2, \ldots, K,\ \bigcup_j Q_j = S; \tag{2.3}$$

$$||Q_i| - |Q_j|| \le T \quad (0 < i \le K, 0 < j \le K); \tag{2.4}$$

$$BOOL(i,j) := \begin{cases} 1, & \text{if } j \in Q_i; \\ 0, & \text{if } j \notin Q_i; \end{cases} \tag{2.5}$$

$$G[j] := \sum_{i=1}^{k} BOOL(i,j) \quad (j \in S); \tag{2.6}$$

$$\sum_{j \in S} G[j] \le 2^P \times Rd \qquad (j \in S); \tag{2.7}$$

$$1 \le G[j] \le K,\ G[j] \in Z^+ \quad (j \in S); \tag{2.8}$$

$$D[k] := \sum_{j \in Q_k} \frac{D\_id[j]}{G[j]} \quad (k = 1, \ldots, K); \tag{2.9}$$

***Minimize:*** $\sum_{i \ne j, i,j=1,\ldots,K} |D[i] - D[j]|.$

Formula (2.3) shows the definition of $Q_j$, the set of ID groups allocated to a specific TCAM chip; (2.4) gives the constraint of the discrepancy of the table sizes among the TCAM chips; (2.5) and (2.6) represent the definition of storing redundancy rate, $G[j]$; (2.7) gives the constrain of the overall redundancy rate; and (2.8) shows that an ID group should have at least one copy (straightforward), while at most $K$ (apparently, it is unnecessary to keep more than one copy in the same TCAM chip).

This problem is proven to be NP-hard. (Please refer to [23] for detailed proofs.) Therefore, we give a heuristic algorithm to solve the problem, as follows:

**Load-Balance-Based Table Construction (LBBTC) Algorithm**

```
Step 1: Pre-Calculation: /* Calculate $G[j]$ iteratively*/
  Define $Rd'$ as the Actual Storing Redundancy;
  Let $a$, $b$ be the lower and upper bound of $Rd'$ in the iteration, respectively.
  $a = 0, b = 2 \times Rd, Rd' = (a+b)/2 = Rd$, $precision = 0.001$.
  Function $F(Rd')$ is defined as
    {$G'[j] = 2^P \times Rd' \times D\_id[j]$        $(j \in S)$;
     $G[j] = \text{Min}\{\lceil G'[j] \rceil, K\}$        $(j \in S)$;
     ($\lceil\ \rceil$ denotes a Ceiling operator here.)
    $Y = \sum_{j \in S} G[j] - 2^P \times Rd$;
      return $Y$; }
  while $F(Rd') \neq 0$ do
    if $F(Rd') < 0$ then $a = Rd'$;
    else $b = Rd'$,
      $Rd' = (a+b)/2$;
    if $|b-a| \le precision$ then break;
  end while;
  $G'[j] = 2^P \times Rd' \times D\_id[j]$        $(j \in S)$;
  $G[j] = \text{Min}(\lceil G'[j] \rceil, K)$        $(j \in S)$;
  Let $W[j] = D\_id[j]/G[j]$        $(j \in S)$;
Step 2: Sort $\{i, i = 1, 2, \ldots, 2^P\}$ in decreasing order of $W[i]$,
and record the results as $\{Sid[1], Sid[2], \ldots, Sid[2^P]\}$;
  for $i$ from 1 to $2^P$ do
    for $j$ from 1 to $G[Sid[i]]$ do
      Sort $\{k, k = 1, \ldots, K\}$ in the increasing order of
      $D[k]$, and record as $\{Sc[1], Sc[2], \ldots, Sc[K]\}$;
    for k from 1 to $K$ do
      if $|Q_{Sc[k]}| < \text{Min}_{i=1\ldots K} |Q_i| + T$ then
        $Q_{Sc[k]} = Q_{Sc[k]} \cup \{Sid[i]\}$;
        $D[Sc[k]] = D[Sc[k]] + W[Sid[i]]$;
        break;
      end for;
    end for;
  end for;
Step 3: output $\{Q_k, k = 1, \ldots, K\}$.
```

The task of this algorithm is to figure out an allocation scheme ($\{Q_k\}$, $k = 1, \ldots, K$) of the ID groups, with which the lookup traffic distribution among the TCAM chips ($D[k]$, $k = 1, \ldots, K$) is as even as possible.

In Step 1, the redundancy rates of the ID groups ($G[j]$, $j \in S$), are pre-calculated aiming at maximizing the sum based on the traffic distribution ($D_id[j]$, $j \in S$), so as to maximize the TCAM space utilization (while satisfying constraint (2.7)). In this step, we adopt the Bisection Method [25], which is simple and quickly converging, to find the solution of $Rd$ when the total redundancy rate of all ID groups equals $2^p Rd$. Then the ID groups are allocated to TCAM chips in Step 2 following two principles: 1) ID groups with larger *partition-load* ($W[j]$) are allocated more preferentially; 2) TCAM chips with lower load are allocated to more preferentially. By applying the greedy algorithm, fairly good result can be achieved when all of the constraints are satisfied. Step 3 outputs the result obtained after Step 2 ultimately.

*2) The Adjusting Algorithm:* Since the LBBTC algorithm is a greedy algorithm, the produced result may still have room for further optimization. The Load-Balance-Based-Adjusting (LBBA) algorithm presented below gives an additional simple but efficient method to further balance the lookup traffic among the TCAM chips.

The primitive idea of this algorithm is that further balanced traffic load distribution can be achieved by exchanging specific ID groups among the TCAM chips, based on the results presented by the LBBTC algorithm and certain principles. This action has no adverse effects on all constraints.

First, we review some concepts. The output of the LBBTC algorithm is a sequence of sets $\{Q_k, k = 1, \ldots, K\}$, with each representing the set of the ID groups allocated to a specific TCAM chip. Corresponding to each of the TCAM chips is the lookup traffic load ratio $D[k]$, $k = 1, \ldots, K$. Now, we divide $\{Q_k, k = 1, \ldots, K\}$ into two sets, say, Set $I$ and Set $J$, $I = \{\text{i} \mid D[i] > 1/K, \text{i} = 1, 2, \ldots, K\}$ while $J = \{\text{j} \mid D[j] \le 1/K, \text{j} = 1, 2, \ldots, K\}$ ($1/K$ represents the average traffic load ratio of the $K$ TCAM chips ).

*Necessary Conditions for Exchanging ID Groups:*
$i \in I, j \in J, n \in Q_i, m \in Q_j, n \notin Q_j, m \notin Q_i$;

$$1)\quad W[n] - W[m] > 0 \tag{2.10a}$$
$$2)\quad D[i] - W[n] > D[j] - W[m]. \tag{2.10b}$$

*Proof of the Optimization:*
Let $D[i]$ and $D'[i]$ denote the traffic load allocated to TCAM $\#i$ before and after the exchange, respectively, then

$$\begin{aligned}
&|D'[i] - D'[j]| \\
&= |(D[i] - W[n] + W[m]) - (D[j] - W[m] + W[n])| \\
&= |(D[i] - W[n]) - (D[j] - W[m]) + W[m] - W[n]| \\
&< |(D[i] - W[n]) - (D[j] - W[m])| + |W[m] - W[n]| \\
&\quad (\Delta = W[n] - W[m] > 0) \\
&= (D[i] - W[n]) - (D[j] - W[m]) + W[n] - W[m] \\
&= D[i] - D[j] = |D[i] - D[j]|.
\end{aligned}$$

□

First, it is straightforward that, under the ***Necessary Conditions,*** the exchange does not break any constraint, then since $|D'[i] - D'[j]| < |D[i] - D[j]|$, it means that the load discrepancy between the two TCAM chips is reduced; we call $Op = |D[i] - D[j]| - |D'[i] - D'[j]|$ the optimized value

TABLE I
TRAFFIC DISTRIBUTION AMONG ID GROUPS

| **ID** | ***0*** | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** |
|---|---|---|---|---|---|---|---|---|
| **D_id%** | 1 | 2 | 3 | 5 | 19 | 7 | 8 | 8 |
| **ID** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** | ***13*** | ***14*** | ***15*** |
| **D_id%** | 6 | 3 | 7 | 2 | 19 | 2 | 2 | 6 |

of the exchanging operation. It represents the optimized value of the objective $\sum_{i\neq j,i,j=1,\ldots,K}|D[i]-D[j]|$ of the LBBTC problem.

***The Load-Balance-Based-Adjusting (LBBA) Algorithm***

```
1) Update Set $I$ and Set $J$;
2) Sort $\{i, i \in I\}$ in the increasing order of $D[i]$, and record the result as $\{I[1], I[2], \ldots, I[|I|]\}$;
   Sort $\{j, j \in J\}$ in the decreasing order of $D[j]$, and record the result as $\{J[1], \ldots, J[|J|]\}$;
   For i from 1 to $|I|$ do /*$|I|$ denotes the number of elements in set $I$*/
      For j from 1 to $|J|$ do
         if $\exists\, n \in Q_i, m \in Q_j$ and
            $n \notin Q_j, m \notin Q_i$ satisfies $W[n]-W[m] > 0$
            and $D[i]-W[n] > D[j]-W[m]$ then
               $Q_i = Q_i - n + m$;
               $Q_j = Q_j - m + n$;
               Go to 1); //Iteration.
```

*Proof of the Convergence of the Algorithm:* We define

$$\Delta = \left\{\delta \middle| \delta = \sum_{n\in P_1\cup P_2} W[n] - \sum_{m\in P_3\cup P_4} W[m],\ \forall P_1,P_2,P_3,P_4 \subset S, \delta > 0\right\}.$$

It is straightforward that set $\Delta$ is independent of the Adjusting Algorithm and $|\Delta| < \infty$. According to the definition of the optimized value $Op$ and Set $\Delta$, it can be easily gotten that $\forall Op$, $Op \in \Delta$, and $\forall Op \geq \operatorname{Min}_{\delta\in\Delta} \delta > 0$. So since the objective

$$\sum_{i\neq j,i,j=1,\ldots,K} |D[i]-D[j]| \geq 0$$

and each exchanging operation reduces the objective by a value larger than $\operatorname{Min}_{\delta\in\Delta}\delta$, the iteration in the algorithm is convergent. □

*An Example:* Suppose that the traffic distribution among ID groups is as given by Table I.

If $Rd = 1.25$, $T = 1$, and $K = 4$, then the allocation result given by the algorithms is as shown in Table II.

TABLE II
PREFIXES ALLOCATION (IN ID GROUPS) RESULTS

| **Group** *ID* | | ***Partition #*** | | | | | **Load ratio** |
|---|---|---|---|---|---|---|---|
| | | *I* | *II* | *III* | *IV* | *V* | |
| **CHIP** | *#1* | 10 | **4** | **12** | 2 | 13 | 24.67% |
| | *#2* | 5 | **4** | **12** | 3 | 0 | 25.67% |
| | *#3* | 7 | **12** | 15 | 1 | 11 | 24.33% |
| | *#4* | 6 | **4** | 8 | 9 | 14 | 25.33% |

## E. Analysis of the Power Efficiency

Traditional TCAM is a fully parallel device. When the search key is presented at the input, all entries are triggered to perform the matching operations, which is the reason for its high power consumption. Current high-density TCAM devices consume as much as 12–15 W per chip when all the entries are enabled for search. However, the power consumption of TCAM can be reduced. We found that not all the entries need to be triggered simultaneously during a lookup operation. The entries that really need to be triggered are the ones matching the input key. Thus, one way of making TCAM power-efficient is to avoid the non-essential entries from being triggered during a lookup operation. In order to measure the efficiency of power consumption in each lookup operation, we introduce the concept of *Dynamic Power Efficiency* (*DPE*), as defined next.

*Definition of Dynamic Power Efficiency:* Let $V_i$ be the number of entries triggered during a specific lookup operation and matching the input search key $i$, and let $W$ be the total number of entries triggered during a specific lookup operation. We define $DPE\,(i)$ as

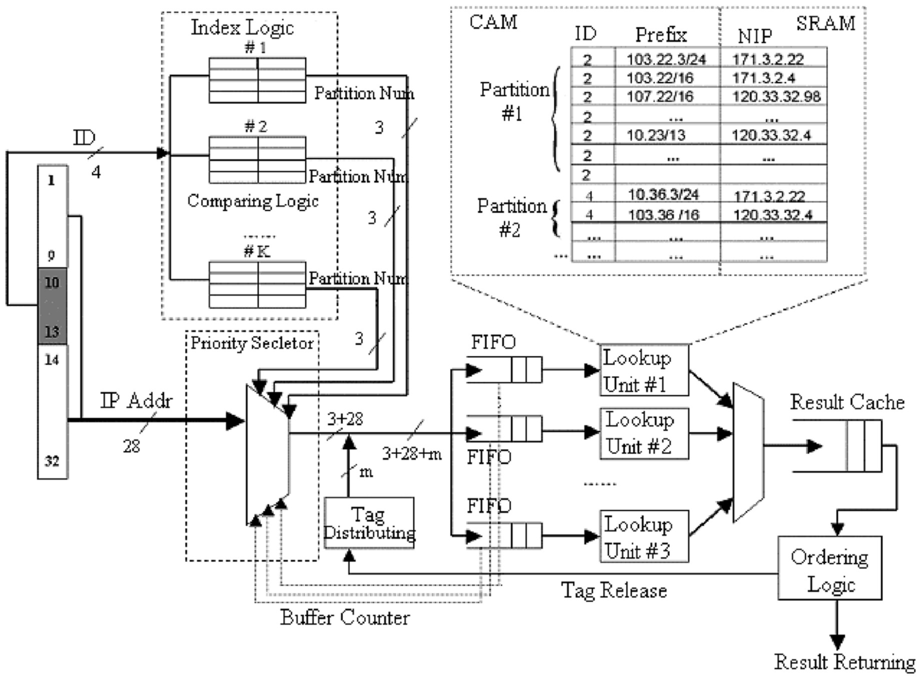
$$DPE\,(i) := \frac{V_i}{W}. \tag{2.11}$$

For instance, suppose that the input search key $i$ is "A.B.C.D", and there are five route prefixes matching this key in the route table, which are: A/8, A.B/12, A.B/16, A.B.C/24, and A.B.C.D/32. The size of the total route table (all in a single chip) is 128,000, and all the entries are triggered during a lookup operation, then

$$DPE\,(i) = \frac{5}{128{,}000} = 3.90625 \times 10^{-5}.$$

There are two ways to decrease the number of entries triggered during a lookup operation. One way is to store the entries in multiple small chips instead of a single large one. The other way is to partition one TCAM into small blocks and trigger only one of them during a lookup operation. If the hardware (TCAM) supports the partition-disable function, the proposed scheme will benefit from both methods.

We continue using the example mentioned above. If we use four small TCAMs instead of (to replace) the large one and partition each of them into eight blocks to store the route prefixes, assuming that the storing redundancy rate is 1.25, then the number of prefixes stored in each of the $4\times 8 = 32$ blocks should be $128{,}000 \times 1.25/32 = 5{,}000$. Applying the proposed TCAM partition method and supposing that the hardware supports the partition-disable feature, the *DPE* is now increased



| ID | Prefix | NIP |
|---|---|---|
| 2 | 103.22.3/24 | 171.3.2.22 |
| 2 | 103.22/16 | 171.3.2.4 |
| 2 | 107.22/16 | 120.33.32.98 |
| 2 | ... | ... |
| 2 | 10.23/13 | 120.33.32.4 |
| 2 | ... | ... |
| 2 | | |
| 4 | 10.36.3/24 | 171.3.2.22 |
| 4 | 103.36 /16 | 120.33.32.4 |
| ... | ... | ... |
| ... | ... | ... |

Fig. 4. Schematics of the complete implementation architecture.

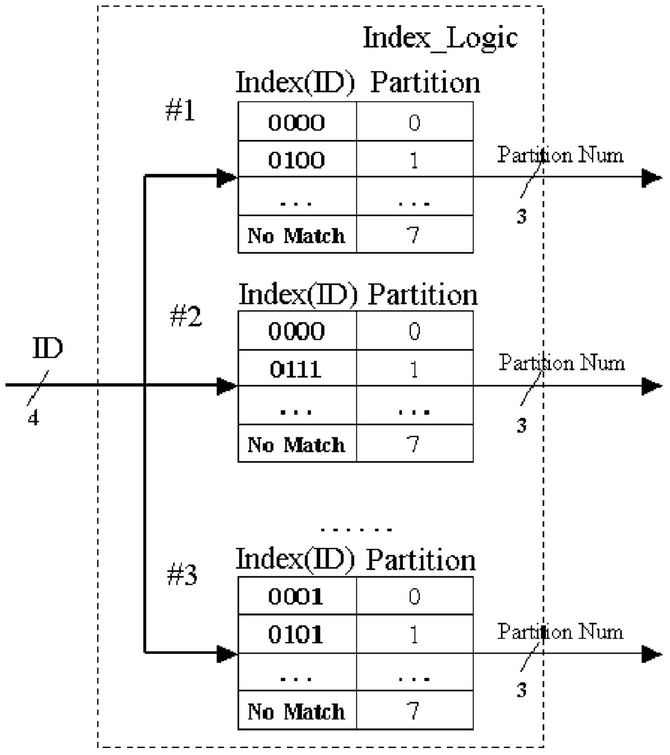
to 5/5000 $=1.0\times10^{-3}$, meaning that the power consumption efficiency is increased by a factor of more than 25.

## III. Complete Implementation Architecture

The complete implementation architecture of the ultra-high-throughput and power-efficient lookup engine is shown in Fig. 4. Given an incoming IP address to be searched, the ID bits of the IP address are extracted and delivered to the index logic for a matching operation. The index logic will return a set of partition numbers indicating which TCAMs and which block inside each TCAM may contain the prefixes matching the IP address. The Priority Selector (i.e., the adaptive load balancing logic) selects one TCAM with the shortest input queue (FIFO) from those containing the matching prefixes and sends the IP address to the input queue corresponding to the selected TCAM. In order to keep the sequence of the incoming IP addresses, a tag (the sequence number) is attached to the IP address being processed.

### *A. The Index Logic*

The function of the Index Logic is to find out the partitions in the TCAMs that contain the group of prefixes matching the incoming IP addresses, as depicted in Fig. 5. It is composed of groups of parallel comparing logics. Each group of comparing logics corresponds to one TCAM while each comparing logic corresponds to a partition in the TCAM. The *Index* field represents the ID of the prefix group and the *Partition* field represents the block in which this group of prefixes is stored. Each group of the comparing logics has a returning port. If the ID bits of the incoming IP address match one of the indexes in a group, the corresponding partition number is returned; otherwise, a partition number "111" (7) will be returned, which represents the no-matching information. Because the data bus width of the



| Index(ID) | Partition |
|---|---|
| 0000 | 0 |
| 0100 | 1 |
| ... | ... |
| No Match | 7 |

| Index(ID) | Partition |
|---|---|
| 0000 | 0 |
| 0111 | 1 |
| ... | ... |
| No Match | 7 |

| Index(ID) | Partition |
|---|---|
| 0001 | 0 |
| 0101 | 1 |
| ... | ... |
| No Match | 7 |

Fig. 5. Schematics of the Index Logic.

comparing logic is narrow and fixed, and only simple "compare" operation is required, it can run at a very high speed.

### *B. The Priority Selector (Adaptive Load Balancing Logic)*

The function of the Priority Selector is to allocate the incoming IP address to the "idlest" TCAM that contains the prefixes matching this IP address, so that the lookup traffic is balanced among the TCAMs adaptively. As mentioned before, we introduce some storing redundancy into our scheme to guarantee the lookup throughput and one prefix may be stored in multiple TCAMs based on the traffic distribution among the ID groups. The Priority Selector uses the counter's status of the input queue for each TCAM to determine which one the current IP address should be delivered to. We give the algorithm of the Priority Selector as follows.

**Algorithm for the Priority Selector**

**Input:**

$Counter[i], i = 1, \dots, K$: Status of the buffer for each TCAM;

$PN[i], i = 1, \dots, K$: Partition numbers of the chips, which are from the Index Logic;

**Output:**

$Obj$: Serial number of the chip that the current IP address should be delivered to.

```
for $i$ from 1 to $K$ do /* To find a feasible solution */
    if $PN[i] \neq 7$
    then $Obj = i$; break;
end for;
for $i$ from $Obj + 1$ to $K$ do
    if ($Counter[i] < Counter[Obj]$) and ($PN[i] \neq 7$)
    then $Obj = i$;
end for;
```

Because the status of queue counters are independent of the current IP address, and high precision is not required here, this component can also be implemented in high-speed ASIC with ease.

### C. The Ordering Logic

Because multiple input queues exist in the proposed scheme, the incoming IP address will leave the result cache (as shown in Fig. 4) in a different sequence from the original one. The function of the ordering logic is to insure that the results will be returned in the same order as that of the input side. An architecture based on Tag-attaching is used in the Ordering Logic. When an incoming IP address is distributed to the proper TCAM, a tag (i.e., sequence number) will be attached to it. Then, at the output side, the Ordering Logic uses the tags to reorder the returning sequence.

### D. An Example

Assuming that the traffic distribution among the ID groups is given as shown in Table I, we use the proposed load-balance-based algorithm to construct the tables in the TCAMs and get the result as shown in Table II. When an IP address, 166.103.142.195, is presented to the lookup engine, its ID "1100" (12) is extracted and sent to the Index Logic. The ID is compared with the 20 indexes in four groups simultaneously. The partition numbers are returned as "010" (2), "010" (2), "001" (1), and "111" (7), which means that TCAM #1, #2, and #3 contain the group of prefixes matching the IP address, while TCAM #4 does not. These results are then sent to the Priority Selector. Suppose that the counter values of the three TCAMs (#1, #2, and #3) are 6, 7, and 3, respectively. TCAM #3 is selected due to its smallest counter value. Then, the IP address 166.103.142.195, the partition number "001", and the current sequence number are pushed into the FIFO queue corresponding to chip #3. When reaching the head of the queue, the IP address is popped out and sent to partition #1 (of TCAM #3) to perform the lookup operation. The final result is sent out by the Ordering Logic in the original order according to its tag attached. All the above processing steps can be carried out in a pipelined fashion.

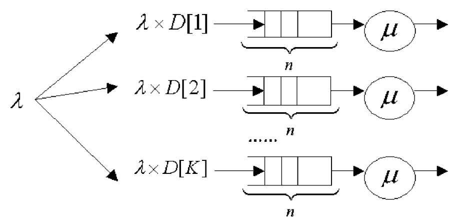


Fig. 6. M/D/1/n queueing model of the proposed mechanism.

## IV. Performance Analysis and Simulation Results

### A. Performance Evaluation (Worst-Case Performance Evaluation)

Statistically high performance may NOT be sufficient for certain real-time or loss/delay sensitive applications. An ultra-high-speed solution should also have deterministic or worst case guaranteed high performance. Though it is fairly difficult to theoretically analyze the performance of the proposed scheme accurately because of the variation of the scheme and the non-determinacy of the lookup traffic, we find a way to estimate its statistical performance lower bound based on certain reasonable assumptions of the lookup traffic.

*Assumption 1:* We use queueing theory to model the lookup engine and assume that the arrival process of the incoming IP addresses is a Poisson process[5] with average arrival rate $\lambda$. The service process of lookup operation follows a deterministic distribution with a constant service rate $\mu$ and service time $T_s = 1/\mu$, which is independent of the arrival process. The buffer of each queue is finite with room for $n$ customers (IP addresses).

Based on Assumption 1 and according to the LBBTC mechanism, the proposed scheme can be modeled as $K$ independently distributed M/D/1/n queueing systems, as shown in Fig. 6. Here we omit the positive effect caused by the adaptive load balancing mechanism, since we are evaluating the statistical performance lower bound.

[5]Although recent studies of Internet network traffic show that the packet arrival process does not follow Poisson distribution any more at the edges of the network but "multi-fractal" or "self-similar" distribution, this phenomenon diminishes within the core of a large network [20]. Since our PC scheme is for core level high-end applications, it is reasonable to use Poisson arrival mode here.

The traffic intensity of each queue is defined as

$$\rho_t = \lambda \times \frac{D[t]}{\mu},\ t = 1, \dots, K. \tag{4.1}$$

Let $\{Q_i\}_{i=1}^{\infty}$ be the (stochastic) process of the number of the IP addresses in the queue at the time of the $i$th arrival, and define

$$\alpha_j(\rho_t) := \sum_{i+m=j-2} \frac{e^{\rho_t(i+1)}(-1)^m \rho_t^m (i+1)^m}{m!}\ (j \geq 2). \tag{4.2}$$

Then the parameters of each queue are given by [18], [19] as:

*The Loss Probability (Blocking Probability):*

$$PL_t := P(Q = n) = \frac{1 + (\rho_t - 1)\alpha_n(\rho_t)}{1 + \rho_t \alpha_n(\rho_t)}; \tag{4.3}$$

*Throughput of the server (TCAM):*

$$\begin{aligned} Thr[t] &= \lambda_t(1 - PL_t) = \frac{\lambda_t \alpha_n(\rho_t)}{1 + \rho_t \alpha_n(\rho_t)} \\ &= \frac{\lambda \times D[t] \times \mu \times \alpha_n\left(\lambda \times \frac{D[t]}{\mu}\right)}{\mu + \lambda \times D[t] \times \alpha_n\left(\lambda \times \frac{D[t]}{\mu}\right)}; \end{aligned} \tag{4.4}$$

*Throughput of the whole system:*

$$Thr = \sum_{t=1,\dots,K} Thr[t] = \sum_{t=1,\dots,K} \frac{\lambda \times D[t] \times \mu \times \alpha_n\left(\lambda \times \frac{D[t]}{\mu}\right)}{\mu + \lambda \times D[t] \times \alpha_n\left(\lambda \times \frac{D[t]}{\mu}\right)}. \tag{4.5}$$

It is straightforward that the more unbalanced the distribution (i.e., $\{D[t], t = 1, \dots, K\}$) is, the lower the lookup throughput of the system can be. To evaluate the worst case performance of the system, we first prove that the traffic load ratio among the TCAM chips, $\{D[t], t = 1, \dots, K\}$, satisfies

$$\frac{1}{K} - \frac{1}{2^P} \leq D[t] \leq \frac{1}{K} + \frac{1}{2^P} \tag{4.6}$$

under the LBBTC algorithm with two sound assumptions.

*Assumption 2:* According to the LBBTC algorithm, given a traffic load ratio of a specific ID group, e.g., ID group $i$, if the load ratio $D_i d[i]$ is large enough, then it would be allocated to all of the $K$ TCAM chips, i.e., $G[i] = K$. In this case, the traffic of this ID group is evenly shared by all the $K$ chips, and will not cause unbalance; so as far as the worst case is concerned, we assume that $G[j] < K, j \in S$ to simplify the proving process.

*Assumption 3:* Practically, when the storage tolerance $T$ (please see Section II for detailed definition) is no less than 2, Constraint 2.4 seems to be always satisfied during the distributing process of the LBBTC algorithm, according to numerous experiments. For the sake of simplifying the proof, we release Constraint 2.4 in the following evaluation.

*Lemma 1:* $\underset{j \in S}{\text{Max}}\, W[j] \leq 1/2^P$.

*Proof:* Please see the Appendix. □

*Lemma 2:* $\underset{k=1,\dots,K}{\text{Max}}(D[k]) - \underset{k=1,\dots,K}{\text{Min}}(D[k]) \leq 1/2^P$.

*Proof:* Please see the Appendix. □

*Theorem 2:* $\{D[t], t = 1, \dots, K\}$ satisfies

$$\frac{1}{K} - \frac{1}{2^P} \leq D[t] \leq \frac{1}{K} + \frac{1}{2^P}.$$

*Proof:* According to Lemma 2,

$$\underset{k=1,\dots,K}{\text{Max}} D[k] - \underset{k=1,\dots,K}{\text{Min}} D[k] \leq \frac{1}{2^P}, \text{ and note that:}$$

$$\underset{k=1,\dots,K}{\text{Min}} D[k] \leq \frac{1}{K} \sum_{k=1,\dots,K} D[k] = \frac{1}{K},$$

$$\underset{k=1,\dots,K}{\text{Max}} D[k] \geq \frac{1}{K} \sum_{k=1,\dots,K} D[k] = \frac{1}{K}, \text{ so :}$$

$$\begin{aligned} &\underset{k=1,\dots,K}{\text{Max}} D[k] - \frac{1}{K} \leq \underset{k=1,\dots,K}{\text{Max}} D[k] \\ &\quad - \underset{k=1,\dots,K}{\text{Min}} D[k] \leq \underset{j \in S}{\text{Max}}\, W[j] \leq \frac{1}{2^P} \\ &\Rightarrow \underset{k=1,\dots,K}{\text{Max}} D[k] \leq \frac{1}{2^P} + \frac{1}{K}. \end{aligned}$$

It can be proved similarly that $\underset{k=1,\dots,K}{\text{Min}} D[k] \geq 1/K - 1/2^P$. □

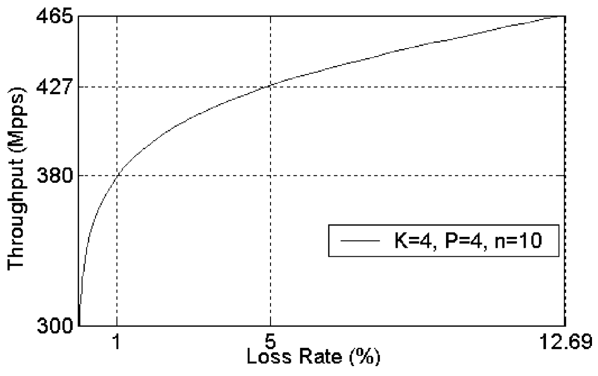
Then, according to (4.5) and (4.6), in the worst case, $K/2$ of the TCAM chips should be allocated with $D[t] = 1/K + 1/2^P$, while the other $K/2$ chips should be allocated with $D[t] = 1/K - 1/2^P$, hence the worst case lookup throughput is given by

$$\begin{aligned} Thr_{\text{worst}} = &\frac{\lambda \times \left(\frac{1}{K} + \frac{1}{2^P}\right) \times \mu \times \alpha_n\left(\lambda \times \frac{\left(\frac{1}{K} + \frac{1}{2^P}\right)}{\mu}\right) \times \frac{K}{2}}{\mu + \lambda \times \left(\frac{1}{K} + \frac{1}{2^P}\right) \times \alpha_n\left(\lambda \times \frac{\left(\frac{1}{K} + \frac{1}{2^P}\right)}{\mu}\right)} \\ &+ \frac{\lambda \times \left(\frac{1}{K} - \frac{1}{2^P}\right) \times \mu \times \alpha_n\left(\lambda \times \frac{\left(\frac{1}{K} - \frac{1}{2^P}\right)}{\mu}\right) \times \frac{K}{2}}{\mu + \lambda \times \left(\frac{1}{K} - \frac{1}{2^P}\right) \times \alpha_n\left(\lambda \times \frac{\left(\frac{1}{K} - \frac{1}{2^P}\right)}{\mu}\right)} \end{aligned} \tag{4.7}$$

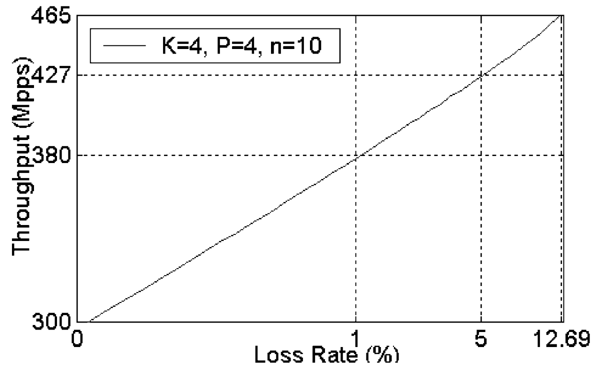
Supposing that four 133 MHz TCAM ($K = 4, \mu = 133$ MPPS) chips are used in the parallel lookup mechanism and the buffer size of each TCAM is 10, i.e., $n = 10$, Fig. 7 depicts the relationship between the lookup throughput and the loss rate according to (4.3) and (4.7). We find that, in the worst case, a statistical throughput of about 427 MPPS can be achieved when the loss rate is restricted within 5%. We also note that by shifting *both* axes to logarithmic scale, the relation curve, as depicted in Fig. 7(b), approximately turns into a line, which constitutes a more intuitive way to estimate their relationship.

## B. Simulation Results

In addition to the theoretical analysis of the algorithm, we have run a series of experiments and simulations to measure its performance and adaptability on different types of traffic load distributions. In the case of four TCAMs with a buffer depth $n = 10$ for each, suppose that the route table is the Mae-West



(a)



(b)

Fig. 7. The relationship between the lookup throughput and the loss rate when adopting four 133 MHz TCAM chips. (a) Linear scale on both axes. (b) Logarithmic scale on both axes.

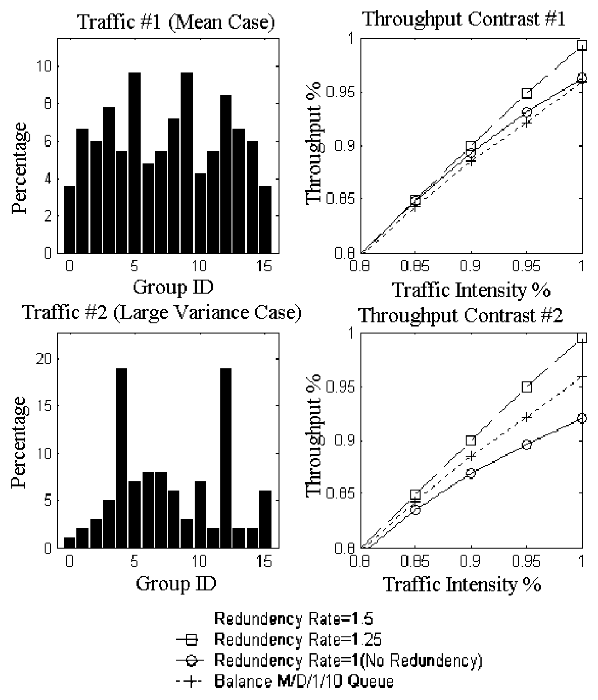


Fig. 8. Comparison of the throughput using different redundancy rate and different traffic distribution.

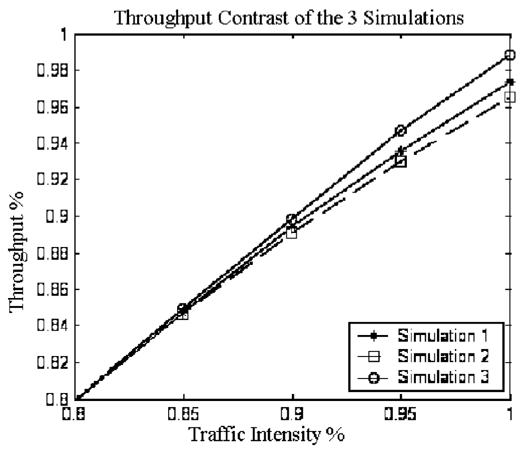


Fig. 9. Results of the three simulations.

table introduced in Section II-B and the arrival processes corresponding to the ID groups are all independent Poisson processes. We use two different traffic load distributions among the ID groups to simulate of the proposed scheme. The results are given in Fig. 8.

When the traffic load is evenly distributed (Case #1, the top two diagrams in Fig. 8), we learn from the simulation result that the introduction of storing redundancy only improves the throughput slightly. The load-balance-based memory organization has already restricted the loss (block) rate within 5%. On the other hand, in the case of skewed distribution (Case #2, the bottom two diagrams in Fig. 8), the storing redundancy improves the lookup throughput distinctly when the system is heavily loaded, even though the redundancy rate is as low as 1.25. In both cases, a redundancy rate of 1.25 guarantees the throughput to be nearly 100%, which means that when four TCAM chips work in parallel, the proposed scheme improves the lookup throughput by a factor of 4 at the expense of only 25% more memory space.

In order to measure the stability and adaptability of the proposed scheme when the traffic distribution varies apart from the original one over time, we run the following simulations with the redundancy rate of 1.25.

1) Assuming that the forwarding table is constructed from the traffic distribution given in Case #2 shown in Fig. 8, we apply the lookup traffic with the distribution given in Case #1. 2) Assuming that the forwarding table is constructed from the traffic distribution given in Case #1 shown in Fig. 8, we apply the lookup traffic with the distribution given in Case #2. 3) Assuming that the forwarding table is constructed from the traffic distribution given in Case #1, we apply strictly even-distributed lookup traffic.

Fig. 9 shows the results of the three simulations. Although the traffic distribution varies a lot, the lookup throughput drops less than 5%, meaning that the proposed scheme is not sensitive to the variation of traffic distribution. In fact, the adaptive load-balancing mechanism plays an important role in such cases.

The input queue and the ordering logic also introduce some processing latency to the incoming IP addresses. Fig. 10 shows simulation results of the queueing and the entire processing (ordering logic included) latency of our mechanism. The entire processing delay is between 9 to $12T_s$ (service cycles). If 133 MHz TCAMs are used, the delay is around 60 ns to 90 ns, which is

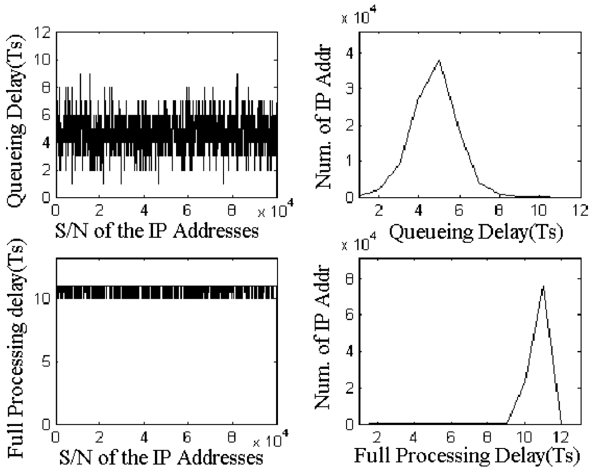


Fig. 10. Processing latency and latency distribution of the incoming IP addresses.

acceptable. We also find that the jitter of the entire processing latency is small, which is critical for hardware implementation.

## C. The Updating Issue

A relatively special point of the updating problem in the proposed scheme is that there are two kinds of reasons leading to TCAM update. One is routing information update (route announcement or route deletion), and the other is traffic load pattern changes.

TCAM table update caused by routing information modification is simply similar to that of the conventional TCAM-based lookup mechanisms. Note that prefixes are organized in ID Groups in the proposed scheme. According to *Theorem 1* (in Section II), in order to ensure Longest Prefix Matching we only need to keep the prefixes in each ID Group stored in decreasing order of their length. It can be easily implemented with incremental update using the algorithm presented in [16]. Also note that for some ID Groups, there may be multiple copies among the TCAM chips, so we need to update all of the copies.

According to the performance evaluation results in the previous subsections, the proposed lookup scheme has a worst case statistical performance lower bound. Hence, if the practical throughput index turns out to be far below the statistical performance lower bound, the traffic load pattern must have changed. This means that the distributed TCAM table should be adjusted (or reorganized) to make it again suitable for the traffic load pattern. This kind of update may be relatively more complex, since it may cause parts of or even the whole distributed TCAM table structure to be reconstructed. However, according to our research, the reconstructing frequency or probability would not be high, because of two main reasons. First, the proposed ultra-high-speed lookup mechanism is targeted at core level applications, where the traffic load pattern should be quite "statistical" in form and relatively stable; second, the proposed scheme is equipped with an effective adaptive load-balancing mechanism, and according to the simulation results (e.g., Fig. 9), the mechanism works very well even when the traffic pattern varies a lot.

TABLE III
A REAL-WORLD EXAMPLE

| Feature | Parameter |
|---|---|
| Number of TCAM Chips | 4 |
| Chip Size | 256K*36b=9Mbit |
| Number of Partitions in Each Chip | 8 |
| Max. Number of Route Entries Supporting | 819.2K |
| Redundancy Rate | 1.25 |
| Working Frequency of TCAM Chips | 133MHz |
| Max. Lookup Throughput | 533Mpps |
| Max. Power Consumption* | $8\times4/8$ =4 Watts |
| High Speed On-chip Cache Requirement | (10*4+40)$\times$32bits=320Bytes |
| Avg. Processing Latency | 75ns |
| Number of Data Buses | 4 |

*: Supposing that, for each chip, the maximum power consumption (all the entries are enabled for search) is 8 Watts.

Nevertheless, we still consider it a non-neglectable issue to rebuild the whole distributed TCAM table though it may only take place in some extreme cases. Fortunately, we find that the Consistent Policy Table Update Algorithm (CoPTUA) for TCAM [24] presented by Z. Wang *et al.* can be employed to solve the problem (note that a route prefix can also be regarded as a 1-D policy classification rule). This algorithm allows the TCAM table to be updated without locking the TCAM and hence not impacting the prefix matching (searching) process.

## V. CONCLUSION

Increasing the lookup throughput and reducing the power consumption of the TCAM-based lookup engine are the two issues discussed in this paper. To distinctly increase the lookup speed and meet the demand of the next generation terabit routers, parallel mechanisms using multiple chips should be deployed. Two topics of applying the parallelism are how to allocate route prefixes evenly and how to balance the lookup traffic among the TCAMs. The power consumption of the TCAM-based lookup engine can be reduced by minimizing the number of entries triggered during each lookup operation. Multi-chip structure and chip partitioning technique are efficient methods for this purpose. In this paper, we proposed a scheme to address both problems. We gave a simple but efficient TCAM table partitioning method and presented a distributed memory organization based on the study of real-world route tables. Then we further developed a mathematical model of the problem on the load-balance-based TCAM table construction and devised a greedy algorithm to solve it. Based on the performance study, given 25% more memory space, the proposed scheme increases the lookup throughput by a factor of 4 (compared with the single-chip scheme) and significantly cut down the power consumption. Table III shows the parameters of a real prototype of the scheme.

## APPENDIX

In this Appendix, we present several lemmas and the corresponding proofs, which are essential to the evaluation of the worst case performance in Section IV.

*Definition:* We define function $F(X)$ as follows, which is useful for the demonstration given later.

Function $F(X)$:

$\{C := 2^P \times Rd'$; (a constant, please see Section II for the corresponding definition)

**Return:** $X/\mathrm{Min}(\lceil C \times X \rceil, K)$;

$\}$

A straightforward observation is that when the variable $X \leq (K-1)/C$, the function value $F(X) \leq 1/C$.

*Lemma 1:* $\underset{j\in S}{\mathrm{Max}} W[j] \leq 1/2^P$.

*Proof:* According to the definition of $W[j]$ (see Section II)

$$\begin{aligned} W[j] &= \frac{D\_id[j]}{\mathrm{Min}(\lceil 2^P \times Rd' \times D\_id[j] \rceil, K)} \\ &= F(D\_id[j]), j \in S. \end{aligned}$$

According to the definition of $G[j]$ and Assumption 2

$$\begin{aligned} &G[j] < K, j \in S \Leftrightarrow \\ &\forall D\_id[j], j \in S, \mathrm{Min}(\lceil 2^P \times Rd' \times D\_id[j] \rceil, K) < K \\ &\Leftrightarrow 2^P \times Rd' \times D\_id[j] \leq K-1 \\ &\Leftrightarrow D\_id[j] \leq \frac{(K-1)}{(2^P \times Rd')} = \frac{(K-1)}{C}. \end{aligned}$$

According to the conclusion drawn in the last section

$$\begin{aligned} \forall j \in S, W[j] &= F(D\_id[j]) \leq \frac{1}{C} \\ &= \frac{1}{2^P} \times Rd' \Leftrightarrow \underset{j\in S}{\mathrm{Max}} W[j] \leq \frac{1}{(2^P \times Rd')}. \end{aligned}$$

According to the definition of $Rd'$ in Section II (Actual Storing Redundancy), $Rd' \geq 1$, so $\underset{j\in S}{\mathrm{Max}} W[j] \leq 1/2^P$. □

*Lemma 2:* $\underset{k=1,\ldots,K}{\mathrm{Max}}(D[k]) - \underset{k=1,\ldots,K}{\mathrm{Min}}(D[k]) \leq 1/2^P$.

*Proof:* We first use mathematical induction to prove

$$\underset{k=1,\ldots,K}{\mathrm{Max}}(D[k]) - \underset{k=1,\ldots,K}{\mathrm{Min}}(D[k]) \leq \underset{j\in S}{\mathrm{Max}} W[j]$$

then according to Lemma 1, we prove

$$\underset{k=1,\ldots,K}{\mathrm{Max}}(D[k]) - \underset{k=1,\ldots,K}{\mathrm{Min}}(D[k]) \leq \underset{j\in S}{\mathrm{Max}} W[j] \leq 1/2^P.$$

We use $t$ to denote the step when allocating ID groups in the LBBTC algorithm. $D_t[k], k = 1, \ldots, K$ denotes the traffic load distributed to the $K$ TCAM chips, after the $t$th step, when ID group Sid$[t]$ is allocated.

A) When $t = 0$, nothing is allocated and so

$$\underset{k=1,\ldots,K}{\mathrm{Max}}(D_0[k]) - \underset{k=1,\ldots,K}{\mathrm{Min}}(D_0[k]) = 0 \leq \underset{j\in S}{\mathrm{Max}} W[j].$$

B) Assume that after the $t$th step, the inequality is still satisfied

$$\underset{k=1,\ldots,K}{\mathrm{Max}} D_t[k] - \underset{k=1,\ldots,K}{\mathrm{Min}} D_t[k] \leq \underset{j\in S}{\mathrm{Max}}(W[j]).$$

C) According to the LBBTC algorithm and Assumption 3, the $(t+1)$th of distributing process is to allocate ID group Sid$[t+1]$ to the $G[Sid[t+1]]$ TCAM chips with the least traffic load distributed. First, we define

$$\underset{k=1,\ldots,K}{\mathrm{Max}} D_t[k] := D_t[n], \underset{k=1,\ldots,K}{\mathrm{Min}} D_t[k] := D_t[m], \text{ and}$$
$$\underset{k=1,\ldots,K}{\mathrm{Max}} D_{t+1}[j] := D_{t+1}[n'], \underset{k=1,\ldots,K}{\mathrm{Min}} D_{t+1}[k] := D_{t+1}[m'].$$

Note that Sid$[t+1]$ must not be allocated to TCAM $\#n$, and it must be allocated to TCAM $\#m$, so

$$D_{t+1}[n] = D_t[n], D_{t+1}[m] = D_t[m] + W[j].$$

1) When $m = m'$, $n = n'$,

$$\begin{aligned} &\underset{k=1,\ldots,K}{\mathrm{Max}} D_{t+1}[k] - \underset{k=1,\ldots,K}{\mathrm{Min}} D_{t+1}[k] \\ &= D_t[n] - D_{t+1}[m] = D_t[n] - D_t[m] - W[j] \leq \underset{j\in S}{\mathrm{Max}}(W[j]), \end{aligned}$$

the inequality is still satisfied after step $t+1$.

2) When $n = n'$, $m \neq m'$, Sid$[t+1]$ must not be allocated to TCAM $\#m'$, so $D_t[m'] \geq \mathrm{Min}(D_t[j]) = D_t[m]$, and

$$\begin{aligned} &\underset{k=1,\ldots,K}{\mathrm{Max}} D_{t+1}[k] - \underset{k=1,\ldots,K}{\mathrm{Min}} D_{t+1}[k] \\ &\qquad = D_t[n] - D_t[m'] \leq D_t[n] - D_t[m] \leq \underset{j\in S}{\mathrm{Max}}(W[j]) \end{aligned}$$

the inequality is still satisfied after step $t+1$.

3) When $n \neq n'$, $m = m'$, note that

$$D_t[n'] \leq \underset{k=1,\ldots,K}{\mathrm{Max}} D_t[k] = D_t[n]$$

so

$$\begin{aligned} &\underset{k=1,\ldots,K}{\mathrm{Max}} D_{t+1}[k] - \underset{k=1,\ldots,K}{\mathrm{Min}}(D_{t+1}[k] \\ &= D_{t+1}[n'] - D_{t+1}[m] \\ &= (D_t[n'] + W[j]) - (D_t[m] + W[j]) = D_t[n'] - D_t[m] \\ &\leq D_t[n] - D_t[m] \leq \underset{j\in S}{\mathrm{Max}}(W[j]). \end{aligned}$$

The inequality is still satisfied after step $t+1$.

4) When $n \neq n'$, $m \neq m'$, Sid$[t+1]$ must be allocated to TCAM $\#n'$, and it must not be allocated to TCAM #m′, namely, $D_{t+1}[m'] = D_t[m'], D_{t+1}[n'] = D_t[n'] + W[j]$; on the other hand, $D_t[n'] \leq D_t[m']$ (according to the allocating principle of the algorithm), and

$$\begin{aligned} &\underset{k=1,\ldots,K}{\mathrm{Max}} D_{t+1}[k] - \underset{k=1,\ldots,K}{\mathrm{Min}} D_{t+1}[k] \\ &= D_{t+1}[n'] - D_{t+1}[m'] \\ &= D_t[n'] + W[j] - D_t[m'] \leq W[j] \leq \underset{j\in S}{\mathrm{Max}}(W[j]). \end{aligned}$$

So the inequality is still satisfied after step $t+1$.

According to 1)–4), the inequality is still satisfied after step $t+1$.

According to A), B), and C), the inequality

$$\underset{k=1,\ldots,K}{\mathrm{Max}} D[k] - \underset{k=1,\ldots,K}{\mathrm{Min}} D[k] \leq \underset{j\in S}{\mathrm{Max}}(W[j])$$

is satisfied after any step of the LBBTC algorithm, and according to Lemma 1

$$\operatorname*{Max}_{k=1,\ldots,K} D[k] - \operatorname*{Min}_{k=1,\ldots,K} D[k] \le \operatorname*{Max}_{j\in S}(W[j]) \le 1/2^P.$$

□

## Acknowledgment

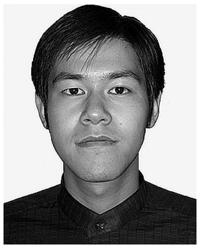
The authors would like to thank Mr. G. Zhang and the reviewers of IEEE INFOCOM 2004 for their valuable comments on this paper. They also thank Dr. X. Wang and Mr. D. Bermingham from DCU, Ireland, for their efforts in proofreading the revised manuscript.

## References

[1] E. Fredkin, "Trie memory," *Commun. ACM*, vol. 3, pp. 490–500, 1960.
[2] D. R. Morrison, "PATRICIA—Practical algorithm to retrieve information coded in alphanumeric," *J. ACM*, vol. 15, no. 4, pp. 514–34, Oct. 1968.
[3] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, pp. 1083–1092, Jun. 1999.
[4] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speed," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 1998, pp. 1240–1247.
[5] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, "Small forwarding tables for fast routing lookups," in *Proc. ACM SIGCOMM*, Cannes, France, Sep. 1997, pp. 3–14.
[6] N.-F. Huang, S.-M. Zhao, and J.-Y. Pan, "A fast IP routing lookup scheme for gigabits switching routers," in *Proc. IEEE INFOCOM*, 1999, vol. 3, pp. 1429–1436.
[7] W. Eatherton, "Hardware-based Internet protocol prefix lookups," M.S. thesis, Dept. Electr. Eng., Washington Univ., St. Louis, MO, 1999.
[8] D. E. Taylor, J. S. Turner, J. W. Lockwood, T. S. Sproull, and D. B. Parlour, "Scalable IP lookup for Internet routers," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 522–534, May 2003.
[9] Cypress. [Online]. Available: http://www.cypress.com/
[10] IDT. [Online]. Available: http://www.idt.com/products/
[11] Netlogic Microsystems. [Online]. Available: http://www.netlogicmicro.com/
[12] H. Liu, "Routing table compaction in ternary CAM," *IEEE Micro*, vol. 22, no. 1, pp. 58–64, Jan.-Feb. 2002.
[13] F. Zane, G. Narlikar, and A. Basu, "CoolCAMs: Power-efficient TCAMs for forwarding engines," in *Proc. IEEE INFOCOM*, San Francisco, CA, 2003, pp. 42–52.
[14] A. J. McAuley and P. Francis, "Fast routing table lookup using CAMs," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1993, pp. 1383–1391.
[15] R. Panigrahy and S. Sharma, "Reducing TCAM power consumption and increasing throughput," in *Proc. HotI'02*, Stanford, CA, p. 107.
[16] D. Shah and P. Gupta, "Fast updating algorithms for TCAMs," *IEEE Micro*, vol. 21, no. 1, pp. 36–47, Jan.-Feb. 2001.
[17] K. Zheng, H. Lu, and B. Liu, "A parallel IP lookup algorithm for terabit router," in *Proc. ICCT*, Beijing, China, Apr. 2003, pp. 478–481.
[18] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[19] S. Alouf, P. Nain, and D. Towsley, "Inferring network characteristics via moment-based estimators," [Online]. Available: http://www-sop.inria.fr/mistral/personnel/Sara.Alouf/Publications/infer.ps.gz
[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
[21] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Beijing, China: The China Machine Press, 1999, pp. 12, 390–391, ISBN 7-111-10921-X.
[22] C. Williamson, "A tutorial on Internet traffic measurement," *IEEE Internet Computing*, vol. 5, no. 6, pp. 70–74, Nov./Dec. 2001.
[23] K. Zheng, C. Hu, H. Lu, and B. Liu, "An ultra high throughput and power efficient TCAM-based IP lookup engine," in *Proc. IEEE INFOCOM*, Apr. 2004, pp. 1984–1994.
[24] Z. Wang, H. Che, M. Kumar, and S. K. Das, "CoPTUA: Consistent policy table update algorithm for TCAM without locking," *IEEE Trans. Comput.*, vol. 53, no. 12, pp. 1602–1614, Dec. 2004.
[25] W. H. Press, *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2002, ch. 9.1, pp. 357–358.

**Kai Zheng** (S'02) received the B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2001. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing.

His research interests include IP address lookup, packet classification and pattern matching associated network security issues.



**Chengchen Hu** (S'04) received the B.S. degree from Northwestern Polytechnical University, Xi'an, China, in 2003. Since 2003, he has been with the Department of Computer Science and Technology, Tsinghua University, Beijing, where he is currently working towards the Ph.D. degree.

His research interests include packet switching, traffic management, and network measurement.

**Hongbin Lu** (S'03) received the B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2002. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing.

His research interests include network security, network measurement, and packet classification.

**Bin Liu** (M'03) received the M.S. and Ph.D. degrees, both in computer science and engineering, from Northwestern Polytechnical University, Xi'an, China, in 1988 and 1993, respectively.

From 1993 to 1995, he was a Postdoctoral Research Fellow in the National Key Laboratory of SPC and Switching Technologies, Beijing University of Post and Telecommunications. In 1995, he transferred to the Department of Computer Science and Technology, Tsinghua University, Beijing, as an Associate Professor, where he mainly focused on multimedia networking including ATM switching technology and Internet infrastructure. He became a full Professor in the Department of Computer Science and Technology, Tsinghua University, in 1999, and is currently the Director of the Laboratory of Broadband Networking Technologies. His current research areas include high-performance switches/routers, high-speed network security, network processors, and traffic management.