Efficient Searching with a TCAM-based Parallel Architecture

Bin Zhang[1], Donghong Qin[2], Xingchun Diao[3], Kun Ding[3], Hao Yan[3]

[1] Peng Cheng Laboratory, Shenzhen, China
[2] School of Information Science and Engineering, GuangXi University for Nationalities, China
[3] Nanjing Telecommunication Technology Research Institute of CESEC, China
bin.zhang@pcl.ac.cn, dhqin@guet.edu.cn, {diaoxch640222, njdingkun}@163.com, yanhao0114@yeah.net

Abstract

Ternary Content-Addressable Memory (TCAM) is a popular hardware device for fast IP address lookup. High link transmission speed of Internet backbone demands more powerful IP address lookup engine. Restricted by the memory access speed, the lookup engine for next-generation routers demands exploiting parallelism among multiple TCAM chips. How to design an efficient engine with high IP lookup speed and less update time while keeping low power consumption is a great challenge in building the next-generation routers. At present, no parallel schemes can make full use of TCAM chips' capability. In this paper, we propose a fast lookup, efficient update and power-saving scheme which can basically make full use of TCAM chips' capability. With N parallel TCAM chips, our proposed scheme can achieve a worst-case speedup factor of (N-1)*90% in cache-update state, and a worst-case speedup factor of N*90% in normal work state. Compared with previous works, our scheme can apparently improve IP lookup performance and update efficiency while keeps low power consumption.

Keywords: Parallel TCAM, Route lookup, Update, Power consumption

1 Introduction

Due to the rapid growth of traffic in the Internet, backbone links of several gigabits per second are commonly deployed. To handle gigabit-per-second traffic rates, the backbone routers must be able to forward millions of packets per second on each of their ports. Fast IP address lookup in the routers, which uses the packet's destination address to determine the next hop for each packet, is therefore crucial to achieve the packet forwarding rates required. Today, many researchers have studied fast lookup schemes for the development of the high performance routers [1]. Most of the schemes can be classified into software approaches based on trie-based and hardware approaches based on TCAM.

Although many alterations have been proposed to optimize the original trie structures [2-4], trie-based IP lookup schemes usually require several memory accesses per lookup and those accesses may be serialized. The lookup speed in trie-based mechanisms using Dynamic Random Access Memory (DRAM) or Static Random Access Memory (SRAM) can hardly be further improved because of its intrinsic characteristic.

For many contemporary hardware architects and system designers Content Addressable Memory (CAM) is a primary choice when it comes to designing high performance lookup systems [5-7]. A specifically interesting type of CAM, called Ternary CAM (TCAM) can store don't-care values in addition to 0s and 1s. This gives TCAM the ability to store variable size pieces of data (called prefixes). TCAM can look up a given key among its contents and find all matching prefixes, all in one clock cycle [8]. Moreover, updating the forwarding table in TCAM-based schemes is generally simpler than that in Trie-based algorithms. Therefore, TCAM have been more and more used in high speed Internet routers in recent years.

However, challenges arises from that (1) the length of IP prefix is variable and the incoming packet does not carry the prefix length information for IP lookup, one IP address may match multiple prefixes in the forwarding table and the longest matching prefix(LMP), should be chosen; (2) advances in fiber-optic technology is pushing the line rate of core routers to 40Gbps or even higher, The state-of-the-art 18Mb TCAM can only operate at a speed of up to 266MHz and performs 133 millions lookup per second [9], barely enough to keep up with the 40Gbps line rate today; (3) the size of the route table has been increasing at a rate of about 10-50k entries per year in the past few years [10]. When IPv6 is widely deployed, even more storage space is needed; (4) frequent updates may consume many computation cycles in the IP lookup engine and result in the degradation of the lookup performance. Efficient update is one of the most important issues together with lookup performance in TCAM-based schemes; (5) the high cost to density ratio and high power consumption are traditionally the major problems in building the lookup engine.

*Corresponding Author: Donghong Qin; E-mail: dhqin@guet.edu.cn
DOI: 10.3966/160792642018121907024

According to the challenges, we classify 3 major concerns in building a forwarding engine based on TCAM: (1) lookup performance; (2) update time; (3) power consumption. We were motivated by the desire to make full use of the TCAM chips free space to increase the lookup throughout as well as achieve fast update and simultaneously reduce power consumption. With this objective, our main work focuses on:

(1) Employing a partitioning technique to evenly distribute the route table entries among the TCAM chips.

(2) Using all free space of every TCAM chip for cache to improve the lookup efficiency.

(3) Updating cache entries efficiently to get a high cache hit rate.

(4) Reducing the update time of route prefixes entries to improve the update efficiency of the whole system.

This work is an extension of our previous work "Bin Zhang, et al., Efficient Searching with Parallel TCAM Chips. The 35th IEEE Conference on Local Computer Networks, LCN 2010, 11-16 Oct. 2010, Denver, Colorado, U.S.A."

The rest of the paper is organized as follows. Related works on parallel schemes of TCAM and our investigation are described in section 2. Section 3 describes the complete architecture of the proposed efficient parallel engine. We present our theoretical performance evaluation in section 4 and simulation results in section 5. We compare the present parallel TCAM lookup schemes in section 6. Finally, we conclude our work in section 7.

2 Related Work and Investigation

Chip-level parallel TCAMs were deployed to circumvent the limitation of a single TCAM, where issues should be appropriately addressed: (1) high memory efficiency, (2) balanced traffic load distribution among parallel TCAM chips, (3) less update time, (4) economical power dissipation.

Many researchers have strived to optimize the TCAM-based lookup engines [11-17]. The partition table is divided by output port into several smaller tables in [11]. The number of tables is equal to the number of output ports on the router. Each table holds a collection of all the entries that map to the output port it corresponds with. Since all entries in a partitioned table map to the same output port, there is no longer a need to keep the entries sorted. When a search occurs, each TCAM looks up the IP address in parallel. Each table outputs the matched lengths to a selection logic. After the selection logic chooses the longest length, the packet is forwarded to the output port based on which table had the longest prefix match.

The basic idea of [12] is to partition all prefixes into different sets based on the relationship among them. For a given destination IP address, the prefixes which have ancestor-descendant relation will be matched simultaneously. The depth of a prefix search trie currently does not exceed 7 even including the default prefix so there can be at most 7 matches. If the forwarding table is partitioned into several TCAMs so that there is no ancestor-descendant relation in each partitioned TCAM, then it is guaranteed that there exists at most one match in each TCAM. The work [11-12] focus only on the update time. The update efficiency is *0(1)* of both schemes, but the lookup performance is poor and power consumption is high.

Panigrahy and Sharma [13] partition all prefixes into 8 different parts equally based on the address range. They put each part in a TCAM chip. When a search occurs, each TCAM looks up the IP address in parallel based on address range. The packet is forwarded to the output port based on which table had the longest prefix match. Hence the lookup performance can be improved. The power consumption can be saved by decreasing the number of the triggered TCAMs access in each lookup operation. CoolCAMs scheme is proposed by [16] to further reduce the power consumption. However, the improved look performance is very limited in [13, 16] due to no further efficient mechanism.

Kai Zheng et al. proposed an architecture in which each TCAM chip had one abundant partition to balance traffic load [15]. It assumed that the lookup traffic distribution among IP prefixes can be derived from the traffic traces. In their optimized system evaluation with simulated traffic, a speedup factor of nearly four can be achieved. However, when the traffic is temporarily biased to a limited number of route prefixes, the multiple selectors will frequently access the same block. The system can only fulfill one of the requests at a time, which drastically hampers the whole throughput.

The more efficient parallel TCAM architecture at present was proposed by Lin et al. [14]. They design a preorder-split method based on the idea of range-base partitioning to evenly distribute the route table entries among the TCAM chips. They analyzed the Internet traffic and observed that the average overall bandwidth utilization was very low but the Internet traffic could be very bursty, so mapping route table partitions into TCAM chips based on long-term traffic distribution [15] cannot effectively balance the workload of individual TCAM chip during bursty periods. After analysis of Internet real traces they found the temporary locality is very strong. Bursts from large TCP flows were the major source of the overall bursty Internet traffic. Nine most common causes of source-level IP traffic bursts exist in Internet, one for UDP and eight for TCP flows. Most of these were due to anomalies or auxiliary mechanisms in TCP and applications. It is indicated that TCP's window-based congestion control itself leads to bursty traffic. As long as a TCP flow cannot fill the pipe between the sender and the receiver, bursts always occur.

Hence, instead of mapping route table partitions into TCAM chips relying on long-term traffic statistics, they used the concept of cache to balance the traffic load adaptively. They used a partition on each TCAM chip as a logical cache for load balancing. A new lookup request was distributed to the TCAM chip with the shortest input queue. There are three different alternatives. (1) If the incoming IP address has been sent to its home TCAM (the TCAM that may contain the matching prefixes), it will get a search operation on the partition and the final result is done. (2) If the incoming IP address has been sent to a non-home TCAM, it will get a search operation on the logical cache. When it is cache-matched, the final result is done. (3) When a cache miss occurs, it will be sent back to the home TCAM directly and case 1) happens again.

In our previous work [17], we propose a crossed address range division and shared caching scheme to balance the traffic on chips, and propose a buddy update method to improve the update efficiency. Our work [17] mainly focuses on update efficiency while keeping the lookup throughput as in [14]. Both schemes [14, 17] are very efficient but both of them still cannot make full use of the parallel TCAM chips' capability. In this work, we take advantage of the all free spaces left in TCAM chips, and improve the previous scheme [14] from six sides:

(1) There is still much free space on each TCAM chip in [14]. Can we use them to increase the logical cache space size? That means we can use all the free space on each TCAM chip as cache to achieve a more efficient lookup performance.

(2) Entries on each logical cache are all same in [14]. In fact, the TCAM chip route entries and the entries of logical cache on that chip may repeat. Hence we need not update the logical cache on that chip when using an entry in the TCAM chip to update caches, which can also save the logical cache space to get a high hit rate.

(3) All TCAM chips have to stop searching when proceeding cache update in [14]. Can we update caches in a pipeline way? We can make sure that there will be N-1 TCAM chips working in parallel during updating cache in this way.

(4) When cache missing occurs a package was sent back to the home TCAM directly in [14], which may lead to dropping packages when bursty traffic hits on one TCAM chip at some time. Can we tag the package with a number counting the times of cache missing? Only when the number exceeds the limit the package can be sent back to home TCAM, otherwise the package is still sent to the TCAM with shortest queuing length. The reason is that cache missing of this cycle does not mean cache missing the next cycle.

(5) Multiple partitions put in one TCAM chip can reduce power consumption using partition-disable technique in [14], but leading to three burdens: (a) Needs to reconstruct all partitions when any partition is full during update; (b) Needs more complex address range arbitration logic; (c) More partitions means more ranges, more ranges means more entries crossing multiple ranges, which weighs update burden. In our scheme, we split the route table into N (denotes TCAM chips number) partitions, and put one partition on one TCAM chip to trade-off update efficiency and power consumption.

We use an efficient update method on each TCAM chip to achieve a worst-case update time O (7) for each TCAM chip. All partitions need to be reconstructed only when any TCAM chip is full during update.

From the above investigations, we can see that the scheme in this work has the following advantages as listed in Table 1.

Table 1. Comparison of schemes

Scheme	Lookup Speed	Update Time	Power Consumption
[11-12]	Not improved	Improved	Not improved
[13, 16]	Limited improved	Not improved	Improved
[15]	Improved	Not improved	Improved
[14, 17]	Improved	Limited Improved	Improved
Our scheme	Highly improved	Highly Improved	Improved

3 Proposed IP Lookup Architecture

3.1 Deciding Partition Method

The first thing in this approach is the partition rule. A good partition rule must meet two primary conditions:

(1) It must create roughly equal partitions;

(2) It must be simple so that the Indexing Logic becomes simple and fast. In other words, determining the target partition by given search key should be possible in one cycle using simple hardware.

Generally three kinds of methods for partitioning the entire routing table were proposed, i.e., key-ID based [15], prefix Trie-based [16] and range-based [13-14] partitioning. The key-ID approach suffered from uneven sub-table sizes and uncontrolled redundancy, which result in a higher memory and power cost. Trie-based partitioning can lower the redundancy and unify sub-table sizes, but it required an extra index TCAM to perform the block selection. As a result, two TCAM accesses are occupied for each lookup request. Range-based partitioning can lower the redundancy and unify sub-table sizes to the most through splitting the routing table into multiple buckets with identical size according to the address range, which is introduced by Panigrahy and Sharma [13] and implemented by Lin et al. using pre-order splitting algorithm [14]. So range-based partitioning can meet condition 1.

Figure 1 illustrates the pipelined structure of the Indexing Logic for TCAM chip selection. It is composed of pairs of parallel comparing logics and an index table. Each pair of parallel comparing logic corresponds to one TCAM chip and is composed of two registers which store the boundary points of each TCAM partition (a partition corresponds to a TCAM chip). Next to the parallel comparing logics is an index table with encoder. The index table which stores the partition distribution information returns a partition number indicating which chip/partition may contain the prefix matching the IP address by the encoded information. Because the data width of the Indexing Logic is fixed and only simple "compare" operation is executed, it can work at very high speed. So range-based partitioning can also meet condition 2. Based on our analysis we choose the range-based partitioning method in [14] to divide the route table.

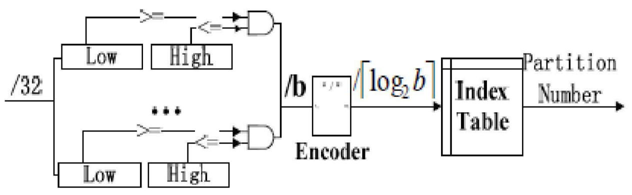

Figure 1. Schematics of the indexing logic

3.2 Logical Cache for Load Balancing

3.2.1 Cache Organization

As mentioned in Section 2, temporal locality of internet traffic is much stronger in the core routers because of the great effect of heavy flow aggregations. To achieve higher lookup throughput, a straightforward design is to deploy a first stage caches working in front of a second stage data TCAMs. An obvious drawback of this conventional approach is that the cache is required to operate at N times the speed of TCAM if there are N parallel TCAMs in the system, which is impractical. Another approach is to add some small extra TCAM chips as caches to work in parallel, but it will increase the complexity of the system structure.

Logical cache was proposed in [14] since there is no additional cache module which implies fewer pins and less packaging cost. Furthermore, employing the existing TCAM cells as logical caches exhibits a better performance cost ratio. We still use logic cache but differently as we mentioned in Section 2. Only a small fixed part of a TCAM chip was used for logic cache in [16], still much free space exists in each TCAM chip unused. Different with the scheme in [14] we use all free space for caching. Our logical cache of each TCAM chip has two parts: the fixed parts and the variant parts, which can significantly increase the cache size.

As depicted in Figure 2, each TCAM chip includes 3 parts: (1) The Route Entry Part is used for prefix entries of a partition. It will gradually enlarge with route table update. When it reaches the bottom line of the fixed cache part and just covers the variant cache part completely, we think the TCAM chip is full. (2) The Variant Cache Part and (3) The Fixed Cache Part are both used for caching. The difference between them is that the fix part is always used for caching and cannot be used for the update of route table entries, which can assure the worst-case speedup factor over N*90% on average with N parallel TCAM chips if the average cache hit rate exceeds 90%. We can get the fixed part size by evaluation to meet the demand. Since the cache entries do not contain these entries in the route entry part, we can confirm that the fixed cache part size is smaller than the logical cache size in [14] when they get the same cache hit rate.

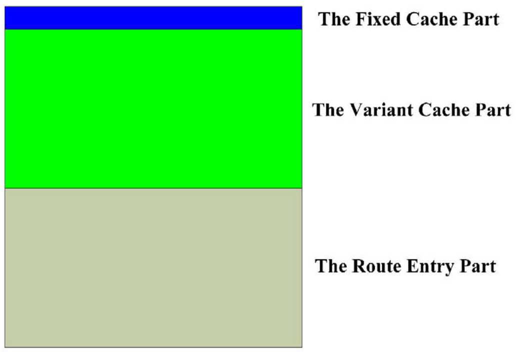

Figure 2. TCAM chip organization

3.2.2 Cache Update

There are two kinds of algorithms for cache mechanisms are proposed in route lookup literatures, caching destination addresses (IPs) [18-20] and caching prefixes [21]. We choose to use the caching prefixes algorithm because it can get the cache hit rate the same as IPs by saving the 97% memories [18-20]. But for caching prefixes, there is one important issue need to be resolved. Suppose p and q are two prefixes where q is a sub-range of p, i.e. p is a prefix of q. If there is a lookup request r redirected to the cache and p is the LMP of r, then p will be added to the cache. However, in some later time if another request r' (whose LMP is q) is redirected to the cache, then the cache will return p as the lookup result (where the correct result should be q). To resolve this problem, we adopt the RRC-ME algorithm [18] in managing the cache. In the above example, the shorter prefix p will be expanded into a longer prefix, based on the Minimal Expansion Prefix (MEP) method in [18]. For instance, there are only two prefixes in a route table, i.e. 1* and 111*. Hence, the MEP for request 1111 is 111*, the MEP for request 1000 is 10*(it has the same next hop as 1*), the MEP for request 1100 is 110*(it has the same next hop as 1*). Since the expanded prefixes are disjoint, there is one and only one possible match result for every input address. Thus, the match result, if any, returned by looking up a cache is valid. Another advantage of prefix expansion method is that any update to the cache only requires one TCAM cycle because the prefixes need not be ordered.

There is another issue about the cache update put forward in [14]. In the proposed cache organization, a TCAM chip is not available to perform IP lookup during cache updates. Hence, a high cache update frequency will seriously affect the system performance. Moreover, immediate cache update in the event of cache missing is very difficult since the system has to evaluate the MEP decomposition of the corresponding prefix. A slow-update mechanism with LRU algorithm was adopted in [14]. Only one cache-missed element is updated within a predefined interval, during this period the other cache-missed elements are ignored. The slow-update mechanism cannot assure the cache hit rate. Our cache update mechanism is described as follows:

(1) The logical cache in each TCAM is filled with prefix entries in other TCAMs' partition at initial state. A register for each TCAM is used to indicate the entries' number of the partition on the TCAM. We increase its value with route table entries' update. We name this kind of register as entry indicator. Its max value equals to TCAM's capability minus the fixed cache capability.

(2) We tag the package with a number counting the times of cache missing. Only when the number exceeds the limit the package can be sent back to home TCAM, which indicates the contention for this entry is very strong. Only in this case we update the caches on other TCAMs.

(3) We update other caches using the cache-missed entry and also its 10 adjacent entries in a pipeline way (The reason is temporal locality of internet traffic is very strong), which can greatly increase the following cache hit rate and can assure N-1 TCAMs work in parallel during cache update.

(4) The match index is instantly available as the home TCAM search results, but the prefix is usually impossible or very slow to read for the commercially available TCAMs. Therefore, to obtain the route prefix the system has to refer to the associated SRAM block. It is assumed that each entry in the SRAM blocks stores the associated prefix together with other forwarding information, besides, each SRAM entry has a flag that indicates whether or not this entry corresponds to a parent prefix. A parent prefix represents a route that has more specific route advertisements (i.e. encompassed prefixes) in the same forwarding table. We adopt the RRC-ME algorithm to deal this situation.

(5) We use a register for each logical cache to indicate the cache update start point every time. We name this kind of register cache indicator. Its initial value equals to the initial value of entry indicator. We can start cache update from this point upward and increase cache indicator's value correspondingly till the value reaches TCAM capability. Cache indicator's value is set back to entry indicator's value when it reaches its highest value.

(6) Route table update on each TCAM chip will make route table entries directly cover the cache entries in variant cache part. When TCAMs finishing route table update the system should check the values of cache indicator and entry indicator. If the entry indicator's value exceeds the cache indicator's value the cache indicator's value is set equal to the entry indicator's value. If the entry indicator's value reaches the max value the system needs to reconstruct all partitions. The two kinds of indicator's value are set equal to new partition entries' number after reconstruction.

Our cache update mechanism is of high efficiency. In our architecture the fixed part size is about 1/10 of the whole TCAM chip, and the initial route table makes each route entry part takes up about half of the whole TCAM chip. The experiment result in Section 5 shows the average cache-update interval is about 8000 clock cycles and the lowest cache hit rate can exceed 96% in this case. The worst case is in TCAM full state, and every route entry part takes 9/10 of the whole TCAM chip. In this case the experiment result shows the average cache-update interval is about 4000 clock cycles and the lowest cache hit rate can exceed 90%.

3.2.3 Proposed Architecture

The detailed implementation architecture of the parallel lookup engine is presented in Figure 3, and the working step can be discribed in Algorithm 1.

Given an incoming IP packet to be searched, the IP address is extracted and delivered to the Indexing Logic (see Figure 1) for a comparison. The Indexing Logic will return a partition number indicating the home TCAM that may contain the matching prefixes. Only when the cache-missing times exceeds the limit (the limit value will be get by test) we set the package can be sent back to the home TCAM, otherwise, the package will be still sent to the TCAM with shortest queuing length. A cache-missing counter tag (initial value is 0) is needed to indicate the cache-missing times. Since multiple input queues and feedback exist in the proposed scheme, incoming IP addresses maybe processed in a non-FIFO order. The Re-ordering Logic maintains the original sequence by using the time stamp attached. The package consists of IP address, partition number (TCAM number), cache-missing counter and time stamp before sent to the adaptive load balance logic.

The package will be sent to the Adaptive Load Balance Logic to decide which TCAMs' FIFO queue should be sent to. The Adaptive Load Balance Logic arbitrates based on two rules: (1) If the cache-missing counter's value exceeds the limit the packet will be sent to the home TCAM directly, otherwise, the packet will be sent to the idlest FIFO; (2) When existing more

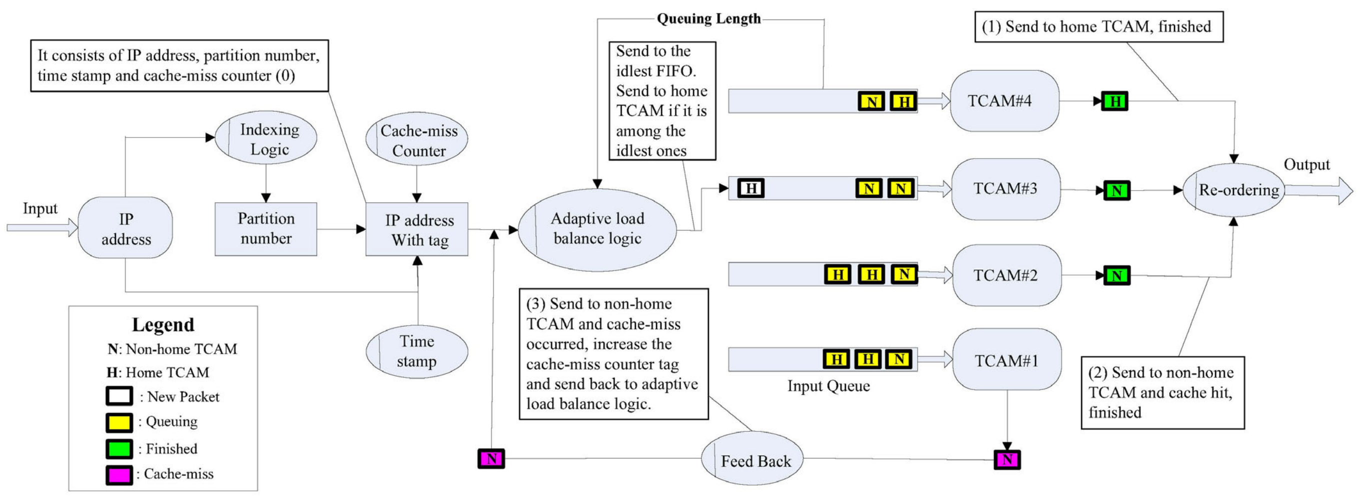
It consists of IP address, partition number, time stamp and cache-miss counter (0)
Queuing Length
Send to the idlest FIFO. Send to home TCAM if it is among the idlest ones
(1) Send to home TCAM, finished
Indexing Logic
Cache-miss Counter
Input
IP address
Partition number
IP address With tag
Adaptive load balance logic
TCAM#4
TCAM#3
TCAM#2
TCAM#1
Re-ordering
Output
Input Queue
Time stamp
(3) Send to non-home TCAM and cache-miss occurred, increase the cache-miss counter tag and send back to adaptive load balance logic.
(2) Send to non-home TCAM and cache hit, finished
Feed Back
Legend
N: Non-home TCAM
H: Home TCAM
: New Packet
: Queuing
: Finished
: Cache-miss
N
H

Figure 3. Schematics of the complete implementation architecture

Algorithm 1. The working flow of our scheme

```
1. An IP packet arrived.
2. The destination IP is extracted and delivered to the Indexing Logic
3. Output a partition number indicating the matching prefixes.
4. The packet further tagged by cache-missing counter and time stamp.
5. The Adaptive Load Balance Logic decides which TCAMs' FIFO queue the packet should be sent to.
   5.1 If the cache-missing counter's value exceeds the limit, the packet→ home TCAM;
   5.2 or, the packet→the idlest FIFO
       5.2.1 More than one idlest FIFO, home TCAM has priority.
       5.2.2 Others have equal opportunity.
   6. TCAM outputs the next hop information for the packet.
   6.1 home TCAM directly outputs the next hop information;
   6.2 Other TCAMs using cache to decide
       6.2.1 Cache match, output the next hop.
       6.2.2 Cache missing, send back to Adaptive Load Balance Logic.
7. The re-ordering logic reorders packets based on timestamp to output.
```

than one idlest FIFO, if the home TCAM is among the idlest ones the package will be sent to the home TCAM, else it will be sent to chip in a random way.

With the feedback mechanism depicted in Figure 3, there are three different alternatives. (1) If the incoming IP address has been sent to its home TCAM, it will get a search operation on the partition indicated by the Indexing Logic and the final result is done; (2) If the incoming IP address has been sent to a non-home TCAM, it will get a search operation on the logical cache. When it is cache-matched (the result is guaranteed by the RRC-ME), the final result is done. (3) If the incoming IP address has been sent to a non-home TCAM, and when a cache miss occurs, it must be sent back to the Adaptive Load Balance Logic after increasing the cache-missing counter's value by 1 via the Feed Back Logic and case 1) happens again.

4 Performance Analysis

4.1 Lookup Performance

The speedup factor of our scheme is apparently N(N indicates all TCAM chips number) in the best case as in [14-15, 17] when all TCAMs work in parallel. The value of the speedup factor in the average case depends on the incoming streams distribution. The authors of [15] use queuing theory to model the lookup engine and assume that the arrival process of the incoming IP addresses is a Poisson process with average arrival rate, which does not conform to real Internet traffic. Lin et al. [14] has calculated that its lower bound of the speedup factor can reach n-1 on the basis that the average hit rate is no less than (N-2)/(N-1). The work [17] proved their speedup factor larger than N-1 for the real traffic.

Since the logical cache on each TCAM chip in our scheme can have much more space than the logical cache in [14] in the same condition, we can assure that the lookup performance can be much better than [14] due to much higher cache hit rate. Moreover, the cache update mechanism of our scheme is more efficient than [14], which can be testified in Section 5 by experiment. Further, the pipeline cache update way can assure N-1 TCAM chips continue searching during cache update.

If N denotes the parallel TCAM chips' number, and P denotes the average cache hit rate. The speedup factor of our scheme is always a little larger than (N-1)*P in cache-update state, and a little larger than N*P in normal work state.

In our architecture we set fixed part size about 1/10 of the whole TCAM chip, and initial route table makes each route entry part takes about half of the whole

TCAM chip. The best condition is in the initial state, which the cache size is the biggest before no new route table entries join in. The experiment result in Section 5 shows the average cache-update interval is about 8000 clock cycles and the lowest cache hit rate can exceed 96% in this case. This means in initial state the speedup factor is over (N-1)*96% during cache-update, and over N*96% otherwise.

The worst case is in the TCAM full state, and every route entry part takes 9/10 of the whole TCAM chip. In this case the experiment result shows the average cache-update interval is about 4000 clock cycles and the lowest cache hit rate can exceed 90%. This means the worst-case speedup factor is over (N-1)*90% in cache-update state, and over N*90% in normal work state.

4.2 Update Efficiency

One TCAM chip is divided into several buckets in [14-15]. The system has to reconstruct when any bucket reaches full even if there are many free spaces in other buckets on the same chip. If the total number of entries is n, TCAM number is x, and the buckets number is y, the update time complexity is O(n/y) for inner bucket update. The system has to reconstruct for inter-bucket update. The system reconstruction will stop the IP lookup operation, drastically hampering the whole system performance. The work [17] use a buddy update algorithm to improve the update efficiency, the update time complexity can get O(n/2x) for inner-chip update and O(2n/x) for inter-chip update on average.

Similar to our work [17], there are 4 TCAM chips in our parallel lookup engine. So the route table should be divided into 4 partitions as Figure 4 illustrated. This can have 3 advantages versus dividing more partitions and putting multiple partitions on one TCAM: (1) Less reconstruction times during update; (2) Less entries crossing multiple ranges; (3) Less compare logics needed in the Indexing Logic (see Figure 1).

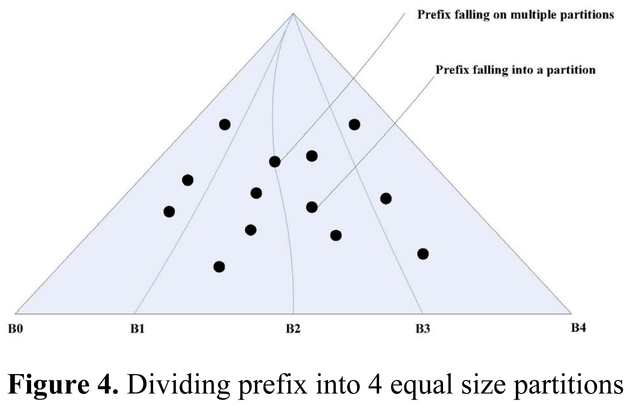
Prefix falling on multiple partitions
Prefix falling into a partition
B0
B1
B2
B3
B4

Figure 4. Dividing prefix into 4 equal size partitions

We use the Preorder-Splitting algorithm in [14] to divide route table into 4 equal size partitions. Let B1, B2, ... denote the boundary points that partition the IP address number line. We place each partition of prefixes in a separate TCAM chip. As for prefixes that fall into multiple ranges, we put them into the chips that correspond to those ranges. It is easy to know that there are at most 32*2 = 64 boundary prefixes that need to be present in a TCAM chip (these prefixes fall into multiple ranges). In fact, the depth of a prefix search trie currently does not exceed 7 even including the default prefix so at most 7*2 = 16 boundary prefixes need to be present in a TCAM chip. This is because any prefix that falls into multiple ranges must be on the path from one of these boundaries to the root. Otherwise, it will strictly lie in the interior of one of the regions carved out by these paths. Each range will need to include, besides prefixes in the interior, the prefixes on the two boundary paths.

During update, two or more TCAM chips need update for a boundary prefix and one TCAM chip need update for an interior prefix. If entry indicator's value of each TCAM chip is less than the max value (TCAM capability menus the fixed cache capability), only chip-level update is needed. Otherwise, we need to reconstruct all partitions using pre-order splitting method. The registers value in the Indexing Logic also needs update according to the new boundary after reconstruction.

Route table update on each TCAM chip will make route table entries directly cover the cache entries in variant cache part as we illustrate in Section 3. As the update on each TCAM chip is individual and does not affect each other, we can use the fast update algorithm proposed by Shah and Gupta [22] on each TCAM chip. Shah and Gupta proposed two fast updating algorithms which are PLO_OPT and CAO_OPT, to reduce the number of memory movements during update. In PLO_OPT all the existing prefixes are sorted by their lengths and free locations are reserved in the middle of the table. Then the number of memory movements per update is no more than L/2 where L is the maximum prefix length, i.e., 32 in IPv4. CAO_OPT exploits the fact that the ordering needs to be maintained only between two prefixes one of which is the prefix of the other. In this algorithm the ordering is referred to as the chain-ancestor ordering (CAO) where a chain means the collection of the prefixes on the path from the root to a leaf node in a prefix search trie. In CAO_OPT the worst case number of memory movements per update has been reduced to D/2 where D is the maximum length of chains. However, free locations are not reserved in the middle of the table on TCAM chip in our scheme (see Figure 2), but they are reserved in front of the table on TCAM chip. Hence, the chip-level update efficiency can achieve an O(D) worst-case update by CAO_OPT algorithm. The depth of a prefix search trie currently does not exceed 7 even including the default prefix so the worst-case update time is about O(7). To support the update operations, the algorithm needs to maintain an auxiliary trie data structure to keep track of the prefixes stored in the TCAM.

4.3 Power Consumption

There are two ways to decrease the number of entries triggered during a lookup operation. One is to store the entries in multiple small chips instead of a single large one. The other one benefits from a new feature of some TCAMs called "partition-disable". The key idea is to split the entire routing table into multiple partitions or buckets, where each bucket is laid out over one or more TCAM blocks. During a parallel lookup operation, only the block(s) containing the prefixes that match the incoming IP address is (are) triggered instead of all the entries in the original table. In this fashion, TCAM's power consumption can be dramatically reduced.

Similar to the work [14-15], Our proposed scheme can benefit from both. If the TCAM chips have not the "partition-disable" feature, assume the power consumption for a TCAM without partitioning search is K, and the TCAM is split into N TCAM chips, let M (M<=N) denotes the TCAM chips number in a working state at a moment. The power consumption should be MK/N at this moment. So the power consumption can be no more than K at any moment.

If the TCAM chips have the "partition-disable" feature, as we mentioned in Section 3 there is a register called entry indicator for each TCAM chip indicating the border of partition area and cache area. We can decide the blocks for each area (maybe a block shared by the two parts at the border) according to the entry indicator. When a package was sent to a home TCAM chip, only the blocks of partition area are triggered. When a package was sent to a non-home TCAM chip, only the blocks of cache area are triggered. Assume the power consumption for a TCAM without partitioning search is K, the blocks number of the TCAM without partitioning is T, and S denotes the blocks number being triggered at a moment. The power consumption should be SK/T at this moment. If the cache area and the partition area almost equal, the power consumption is about K/2 when the lookup engine works at full speed.

5 Experiments and Simulations

In addition to the theoretical analysis we have run a series of experiments and simulations to measure the scheme performance. Table 2 shows the parameters of a real implementation of our scheme. Our scheme is flexible and scalable, and can be easily extended for IPv6 address lookup. Yang et al. [23] discuss how to extend for IPv6 address lookup.

Table 2. Parameters in the test scheme

Feature	Parameter
Number of TCAM Chips	4
Chip Size	32K*36b
Percentage of the Fixed Cache	10%
Max. Number of Route Entries	Supporting 115.2K
Working Frequency of TCAM Chips	266MHz
Max. Lookup Throughput	1.064Gbps
Max. Power Consumption(rare)	5*4*90%=18W (5W/per chip)
Number of Data Buses	4

The route entries are collected from backbone router of the China Education and Research Network (CERNet). The reason we use 32K*32b chip size (one can decide the chip size based on the real situation, usually bigger than this size) is that the largest route entries number is about 120K. We must make all TCAM chip full for the worst-case test so the max router entries number of our system cannot exceed 120K. To get about half of the whole chip space for cache on each chip in the initial state, we select randomly about 57.6K entries from 120K entries and partition them using pre-order splitting algorithm in [16]. For the worst case (TCAM full state) we use about 115.2K entries from 120K entries and partition them using pre-order splitting algorithm. Table 3 and Table 4 show the partitioned results of the two cases respectively.

Table 3. Partition result in the initial state

Partition	Range Low	Range High	Size
1	0.0.0.0	70.89.255.255	16745
2	70.90.0.0	128.56.78.255	16745
3	128.56.79.0	192.65.132.255	16745
4	192.65.133.0	255.255.255.255	16747

Table 4. Partition result in the TCAM full state

Partition	Range Low	Range High	Size
1	0.0.0.0	65.156.220.255	29491
2	65.156.221.0	112.114.255.255	29491
3	112.117.0.0	145.186.158.255	29491
4	145.186.159.0	255.255.255.255	29492

Before the performance test there are two parameters to be decided. One is the input queue's depth. The other is the limit of cache-missing times. In initial state the comparisons of the parameters with different buffer depth and different limit are shown in Figure 5 to Figure 8. Trading off between the packet loss probability and the system response time, we choose 10 as a typical value of buffer depth and 3 as the limit value of cache-missing times. From Figure 7 we can see the trace sampled from 1Gbps line can be forwarded by our system without packet loss and with high system response time when choosing the two values.

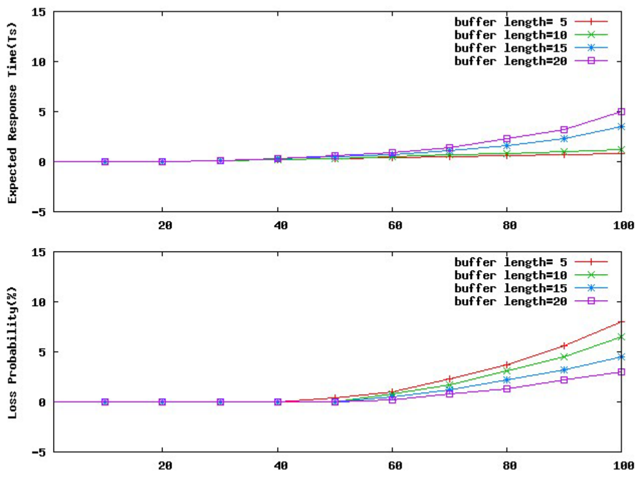

Figure 5. Parameters comparison with different buffer depth when the cache-miss times limit value equals 1

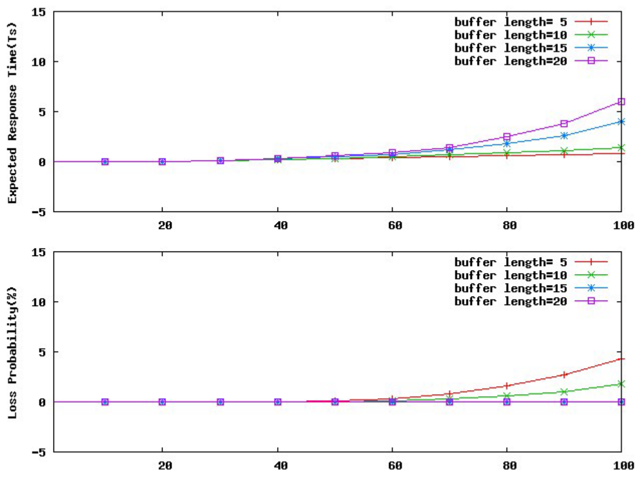

Figure 6. Parameters comparison with different buffer depth when the cache-miss times limit value equals 2

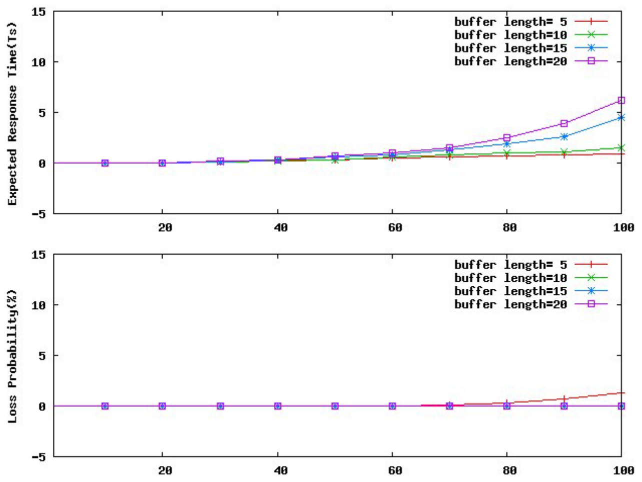

Figure 7. Parameters comparison with different buffer depth when the cache-miss times limit value equals 3

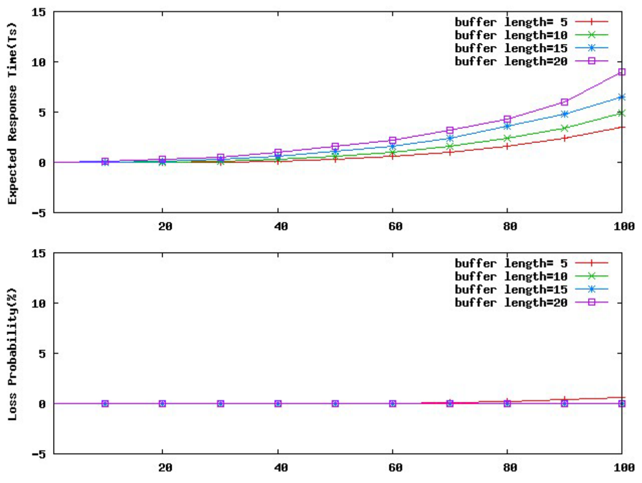

Figure 8. Parameters comparison with different buffer depth when the cache-miss times limit value equals 4

The trace for our test is sampled from the core router located in the network center of Tsinghua University which belongs to Beijing Regional Network Center. It operates at 1Gbps of the Ethernet link bandwidth. The trace is collected from 9:00 to 10:00 on Feb. 21, 2009 and only outbound packets were recorded. The comparisons of the cache-hit rate between the scheme in [14] and our scheme in initial state and in cache-full state (the worst case) are shown in Figure 9 and Figure 10 respectively. The cache-update interval changes according to the cache hit rate in our scheme. The higher cache hit rate, the longer update interval. Figure 9 shows the lowest cache hit rate can exceed 96% in initial state after about 3 seconds in our scheme and the average cache-update interval is about 8526 clock cycles. We choose 8,526 clock cycles as the cache-update interval for the scheme in [14] for comparison. We can see from Figure 9 that our scheme is far superior to the scheme in [14]. The reason is that there are more cache space and more efficient cache-update mechanism guaranteed. The cache hit rate is not 0 at the start versus the scheme in [14] because the cache in each TCAM is filled with route table entries on other TCAMs' partition in initial state. Figure 10 shows the average cache-update interval is about 5,112 clock cycles and the lowest cache hit rate can exceed 90% in initial state after about 4 seconds in our scheme. We choose 5,112 clock cycles as the cache-update interval for the scheme in [14] for comparison. We can see from Figure 10 that our scheme is still superior to the scheme in [14]. One reason is that the cache-update mechanism is more efficient in our scheme for the same cache size. Another reason is that the cache needs not hold the redundant entries of in the same TCAM chip. The result of the two figures shows the more free space in each TCAM, the more superior of our scheme versus [14].

The comparisons of the lookup throughout between the scheme in [14] and our scheme in initial state and in cache-full state (worst case) are shown in Figure 11 and Figure 12 respectively. We can see from the two figures that our scheme can support better for the 1Gbps trace than the scheme in [14], especially in initial state.

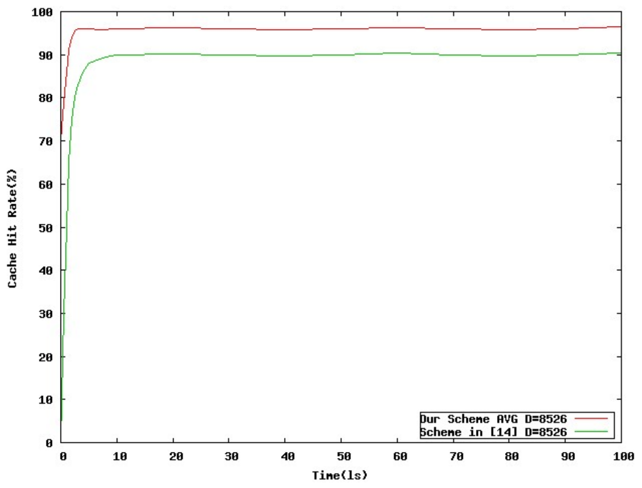

Figure 9. Cache hit rate comparison in initial state

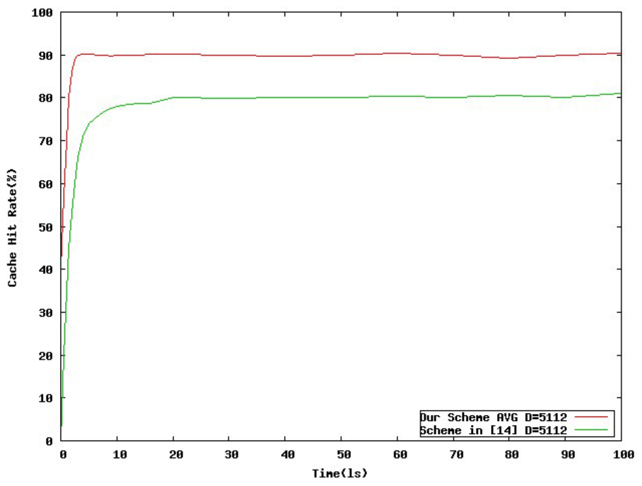

Figure 10. Cache hit rate comparison in TCAM full state

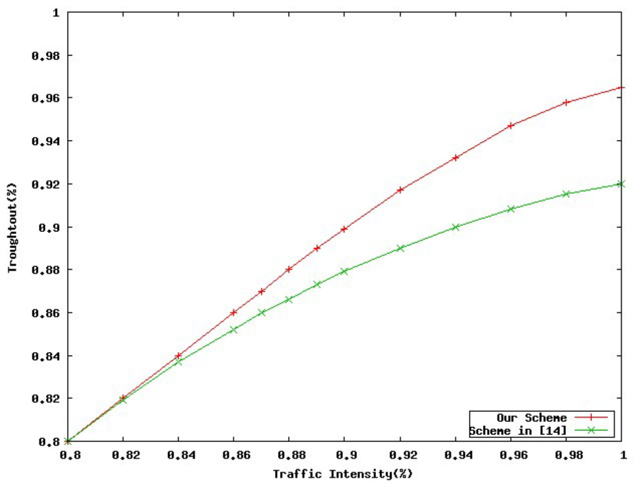

Figure 11. Throughput comparison in initial state

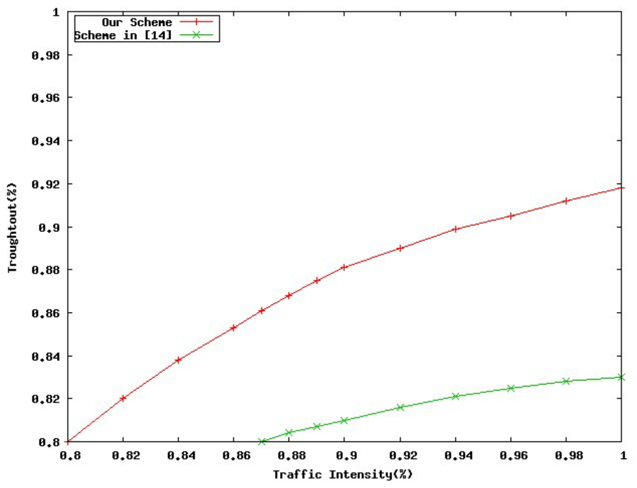

Figure 12. Throughput comparison in TCAM full state

6 Performance Comparison

In this section, we compare our scheme with the schemes of [14-15, 17] in three aspects: (1) lookup performance; (2) update time; (3) power consumption. Table 5 lists the parameters in performance comparison, and table 6 give a detailed performance comparison with scheme [14-15, 17].

Table 5. Parameters in Performance Comparison

Parameter	Concrete Meaning
n	the parallel TCAM chips number
K	the power consumption for a TCAM without partitioning search
m	the working state TCAM chips number
t	the whole TCAM block number
s	triggered block number when searching
H	total entries num
y	buckets num

Table 6. Performance Comparison

Scheme	IP Lookup	Update time	Power consumption
[15]	<n - 1	O(H/y)	Ks/t
[14]	>(n-2)/(n-1)	O(H/y)	Ks/t
[17]	> n - 1	O(H/2n)	Km/n
Our scheme	90%*n	O(7)	Ks/t

In IP lookup performance respect, from table 6 we can see that our scheme is superior to schemes [14-15, 17] when n<10. In real implementation, for practice, the parallel TCAMs' number will not too large. Hence, our scheme is superior to other schemes [14-15, 17] in practice.

In update time respect, since the total entries number is much larger than the TCAM chips number and bucket number, our scheme is far superior to schemes [14-15, 17].

In power consumption respect, our scheme is equivalent to [14-15] and superior to [17]. The scheme [17] cannot use the "partition-disable" feature of

TCAM. For a lookup operation, a chip instead of a partition in a chip must be triggered.

From the above comparison, we can see that our scheme improves both lookup performance and update efficiency while keeps a low power consumption. The basic reason is that our scheme can make almost full use of the free space in the chips to improve efficiency.

7 Conclusions

To distinctly increase the lookup speed and meet the demand of the next-generation routers, parallel mechanism using multiple chips should be deployed. In this paper, we proposed a fast lookup, efficient update and power-saving parallel TCAM-based lookup engine to satisfy the growing demand now. Our key contributions for the parallel TCAM-based lookup engine are as follows:

- Proposed a scheme that can almost make full use of the free space of parallel TCAM chips, which never has been proposed before.
- Proposed an efficient cache-update mechanism.
- Used the state of art mechanisms efficiently to improve the TCAM entries' update and reduce the power consumption.

Compared with previous parallel IP lookup schemes, our schemes can apparently improve the IP lookup performance and update efficiency while keeps a low power dissipation. The scheme is easy to implement and scalable and flexible.

Acknowledgments

The authors would like to express their appreciations to Dong Lin et al. for their sharing the code in their work for us. This work was supported by the Peng Cheng Laboratory network technology simulation verification platform No. PCL2018KP004, the Independent Controllable Ecological Environment No. PCL2018KP005, the National Natural Science Foundation of China under Grant No. 61462009, the China Postdoctoral Science Foundation under No. 2015M582832, the Jiangsu Post-doctor Research Fund of China under No. 1402138C, the Natural Science Foundation of Guangxi under Grant No. 2018JJA17014.

References

[1] M. A. Ruiz-Sanchez, E. W. Biersack, W. Dabbous, Survey and Taxonomy of IP Address Lookup Algorithms, *IEEE Network*, Vol. 15, No. 2, pp. 8-23, March/April, 2001.

[2] P. Gupta, S. Lin, N. McKeown, Routing Lookups in Hardware at Memory Access Speeds, *IEEE INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, CA, 1998, pp. 1240-1247.

[3] N. F. Huang, S. M. Zhao, J. Y. Pan, C. A. Su, A Fast IP Routing Lookup Scheme for Gigabits Switching Routers, *IEEE INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, New York, NY, 1999, pp. 1429-1436.

[4] D. Yu, B. C. Smith, B. Wei, Forwarding Engine for Fast Routing Lookups and Updates, *IEEE Global Telecommunications Conference, GLOBECOM '99*, Vol. 2, Rio de Janeireo, Brazil, 1999, pp. 1556-1564.

[5] T. Pei, C. Zukowski, Putting Routing Tables in Silicon, *IEEE Network Magazine*, Vol. 6, No. 1, pp. 42-50, January, 1992.

[6] A. McAuley, P. Francis, Fast Routing Table Lookup Using CAMs, *IEEE INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future*, San Francisco, CA, 1993, pp. 1382-1391.

[7] P. C. Lekkas, *Network Processors - Architectures, Protocols, and Platforms*, McGraw-Hill, 2004.

[8] V. Srinivasan, B. Nataraj, S. Khanna, *Methods For Longest Prefix Matching In a Content Addressable Memory*, US Patent 6,237,061, January 1999.

[9] CYRESS, http://www.cypress.com/.

[10] G. Huston, Route Table Analysis Reports, http://bgp.potaroo.net/.

[11] E. Ng, G. Lee, Eliminating Sorting in IP Lookup Devices using Partitioned Table, *16th IEEE International Conf. on Application-Specific Systems, Architecture and Processors*, Samos, Greece, 2005, pp. 119-126.

[12] J. Kim, J. Kim, An Efficient IP Lookup Architecture with Fast Update Using Single-Match TCAMs, *6th International Conference on Wired/Wireless Internet Communications (WWIC 2008)*, Tampere, Finland, 2008, pp. 104-116.

[13] R. Panigrahy, S. Sharma, Reducing TCAM Power Consumption and Increasing Throughput, *Proceedings 10th Symposium on High Performance Interconnects*, Stanford, CA, 2002, pp. 107-112.

[14] D. Lin, Y. Zhang, C. Hu, B. Liu, X. Zhang, D. Pao, Route Table Partitioning and Load Balancing for Parallel Searching with TCAMs, *21st IEEE International Parallel & Distributed Processing Symposium*, Long Beach, CA, 2007, pp. 1-10.

[15] K. Zheng, C. Hu, H. Liu, B. Liu, An Ultra High Throughput and Power Efficient TCAM-based IP Lookup Engine, *IEEE INFOCOM 2004*, Vol. 3, Hong Kong, China, 2004, pp. 1984-1994.

[16] F. Zane, G. Narlikar, A. Basu, CoolCAMs: Power-Efficient TCAMs for Forwarding Engines, *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1, San Francisco, CA, 2003, pp. 42-52.

[17] B. Zhang, J. Yang, J. Wu, Q. Li, D. Qin, An Efficient Parallel TCAM Scheme for the Forwarding Engine of the Next-generation Router, *IFIP/IEEE Int'l Symposium On Integrated Network Management*, Dublin, Ireland, 2011, pp. 454-461.

[18] W. L. Shyu, C. S. Wu, T. C. Hou, Efficiency Analyses on Routing Cache Replacement Algorithms, *2002 IEEE*

International Conference on Communications. Conference (ICC 2002), Vol. 4, New York, NY, 2002, pp. 2232-2236.

[19] T. Chiueh, P. Pradhan, High-Performance IP Routing Table Lookup Using CPU Caching, *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, New York, NY, 1999, pp. 1421-1428.

[20] B. Talbot, T. Sherwood, B. Lin, IP Caching for Terabit Speed Routers, *Global Telecommunications Conference on Seamless Interconnection for Universal Services (GLOBECOM '99)*, Vol. 2, Rio de Janeireo, Brazil, 1999, pp. 1565-1569.

[21] M. J. Akhbarizadeh, M. Nourani, Efficient Prefix Cache for Network Processors, *12th Annual IEEE Symposium on High Performance Interconnects*, Stanford, CA, 2004, pp. 41-46.

[22] D. Shah, P. Gupta, Fast Updating Algorithms for TCAMs, *IEEE Micro*, Vol. 21, No. 1, pp. 36-47, Januay/February, 2001.

[23] T. Yang, G. Xie, Y. Li, Q. Fu, A. X. Liu, Q. Li, L. Mathy, *Guarantee IP Lookup Performance with FIB Explosion, Proceedings of SIGCOMM*, Chicago, IL, 2014, pp. 39-50.

Biographies

Bin Zhang received his Ph.D. degree in Department of Computer Science and Technology, Tsinghua University, China in 2012. He received the Bachelor degree in computer software from Zhengzhou University, China in 1998 and the Master's degree in network information security from Shanghai Jiaotong University, China, in 2005. During his Ph.D. career, he publish more than 20 papers in refereed international conferences (NOMS, IM, LCN, IWQoS, APNOMS, etc) and journals (the Computer Journal, JCST, Journal of Software). He is a post doctor in Nanjing Telecommunication Technology Institute, Nanjing, China. His current research interests focus on Internet architecture and its protocols, IP routing technology, network measurement, network management, etc.

Donghong Qin received his Ph.D. degree in Department of Computer Science and Technology, Tsinghua University, China in 2013. He publish more than 20 papers in refereed international conferences and journals. He is a professor in School of Information Science and Engineering, GuangXi University for Nationalities, Nanning, China. His current research interests focus on Internet architecture and its protocols, IP routing technology, network management, etc.

Xingchun Diao is the headmaster and also a professor in Nanjing Telecommunication Technology Institute, Nanjing, China. He has published more than 100 articles in refereed international conferences and journals, and two books on data quality and management. Xinchun's research interests include big data processing, data quality, network measurement, network management, etc. He also serves as a TPC member for several international conferences.

Kun Ding is a senior engeer in Nanjing Telecommunication Technology Institute, Nanjing, China.. He has published more than 20 articles in refereed international conferences and journals. Ding's research interests include network measurement, network management, network traffic anomaly detection, network topology discovery, etc.

Hao Yan is a senior engeer in Nanjing Telecommunication Technology Institute, Nanjing, China. He has published more than 30 articles in refereed international conferences and journals. Hao's research interests include data quality, network management, network security, Internet architecture and its protocols, IP routing technology, etc.