EASTFILE: AN ENERGY AWARE, SCALABLE, AND TCAM BASED FAST IP LOOKUP ENGINE

Hamidreza Mahini[1] and Reza Berangi[2]

[1]Islamic Azad University, Gorgan Branch, Department of Computer Engineering, Gorgan, Iran
hamidreza.mahini@gorganiau.ac.ir

[2]Iran University of Science and Technology (IUST), Department of Computer Engineering, Tehran, Iran
rberangi@iust.ac.ir

ABSTRACT

Routers are one of the important entities in computer networks especially in the Internet. Forwarding IP packets is a valuable and vital function in the Internet routers. Routers extract destination IP address from packets and lookup those addresses in their own routing table. This task is called IP lookup. Ternary Content-Addressable Memories (TCAMs) are becoming very popular for designing high-throughput address Lookup engines on routers: they are fast, cost effective and simple to manage. Despite their premise on high-throughput, large TCAM arrays are prohibitive due to their excessive power consumption and lack of scalable design schemes. Therefore high power consumption in TCAMs is the mainspring of their failure in scalable IP lookup engines. This paper presents a novel and intelligent energy aware TCAM-Based IP lookup engine (EASTFILE) which is scalable for large IP lookup tables and also IPv6 lookup. Energy aware means that power consumption in this scheme is bounded. Actually this method decreases power consumption nearly 74% rather than the referenced architecture in the worst case. Also this method decreases 46% number of match lines in the first step of lookup operation rather than the other multi level enabling techniques or simple partitioning technique. This fact provides using TCAMs in large IP lookup tables and IPv6 addresses. In addition, the main operations of an IP lookup engine such as lookup, insert, withdrawal and update are very efficient in the proposed approach.

KEYWORDS

High Performance Routers, IP Lookup, Ternary Content Addressable Memory (TCAM), Low-Power Design

1. Introduction

The Internet traffic is rapidly growing not only because of the increase of users but also because of the various applications such as multimedia. Therefore, fast link speed and routers are required. The basic routing function of routers is to determinate the next-hop address of each incoming packet based on the packet's destination IP address, named IP lookup. IP lookup is based on IP lookup table. Simply, IP lookup table has two columns; one for storing prefix of IP addresses and the other for saving next hop. Actually IP lookup table is many to one mapping between destination IP address of incoming packets and next hop. Using prefixes instead of IP addresses help us to decrease the table rows which are important in different aspects because the decrement of table rows leads to decreasing memory accesses in software methods and decreasing power consumption in hardware ones. Many minimization algorithms have been presented for generating prefixes from IP addresses such as Espresso-II. Whenever using Prefixes instead of IP addresses, longest prefix matching (LPM) will be arisen.

Generally IP lookup methods divide into three categories: software, hardware and hybrid approaches. In software methods, a data structure, named trie is used. In computer science, a trie, or prefix tree, is an ordered tree data structure that is used to store an associative array where the keys are usually strings. The main problem with software methods is the large number of memory accesses to meet the necessary operations. Moreover, most of them [1–4] can hardly be scaled to support IPv6, because their lookup time grows linearly with the search key (Destination IP Address) length. Some algorithms required constant search time for IPv4, such as DIR-21-3-8 [5]. However, their high storage requirements allow them to be implemented only with mass-but-slow SDRAMs. What's more, the exponentially increased memory requirements with the search key length make them unsuitable for IPv6 implementation.

Hardware approaches usually use TCAMs for implementing lookup table. In hardware methods, power consumption is the mainspring due to their unsuccessfully. One of the decrementing power consumption mechanisms is based on lookup table partitioning. Although these methods are useful, they are not suitable for large tables and IPv6 addresses.

Hybrid methods try to combine the benefits of software with hardware methods. Caching super nodes is a hybrid approach which is attractive in computer network and communication researchers.

A good IP (v6) address lookup scheme should have the following features [6]: (1) High enough lookup throughput to support high-speed interface even in the worst case (i.e., minimum-sized packets coming in back-to-back); (2) small memory requirement, making it practical to be implemented with small but fast memory chips or on-chip caches; (3) scalability with the key length, maintaining the lookup time, and memory requirement at a feasible low level when migrated to IPv6; (4) Low update cost, to cope with the instability of the BGP protocol.

In this paper we present a novel hardware approach which has three important features as follows: being energy or power consumption aware, scalable for IPv6, and fast operation. Ergo this approach called EASTFILE which means energy aware, scalable, and TCAM based fast IP lookup engine. Also performance of the proposed approach has been evaluated on real IPv4 and IPv6 lookup table data which had led to some interesting results.

The rest of the paper is organized as follows. In section 2, we list a number of related works and Section 3 describes the EASTFILE architecture. Section 4 presents the performance evaluation of the proposed scheme, as well as the comparison with other schemes. In section 5, experimental results are presented. Finally, a conclusion and future works is drawn in Section 6.

2. Related Works

As mentioned in the last section, IP lookup techniques are divided into software, hardware and hybrid methods. Software methods are usually based on trie data structure and the large memory access is the main reason of their failure.

The Lulea [6] algorithm, for the first time, develops the concept of bitmap compression to improve storage efficiency, achieving both small memory requirement and almost constant lookup time. Hence much faster SRAM chips can be employed to replace SDRAMs. However, its specially designed memory organization is un-scalable for IPv6 or large route tables. Furthermore, it is known to be very hard to update since it needs leaf-pushing. The Eatherton's Tree Bitmap algorithm [8] 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 therefore supports fast incremental updates. An implementation of the Tree Bitmap algorithm, referred to as Fast IP Lookup (FIPL) [9], shows that a storage requirement of about 6.3 bytes per prefix and performance of over one million lookups per FIPL engine can be achieved. However, FIPL uses a fixed lookup stride of four at each level, hence for each IP address lookup it may require more than 32/4 = 8 memory accesses in the case of IPv4, and 128/4 = 32 in the case of IPv6, which make it also infeasible for IPv6 migration.

The other good software method is DVSBC-PC which was presented by Kai Zheng and his colleagues [6]. DVSBC-PC is abbreviation for Dynamic Variable Stride Bitmap Compression and Path Compression. Although this approach combines two techniques to achieve efficient trie for lookup scheme, but it needs 7.17 memory access for IPv4 lookup in average case. This method provider combined it with TCAM and presented a new approach named DVSBC-PC-CAM. This is one instance of hybrid approaches. With this combination they achieved to 4.33 memory access for IPv4 lookup in average case. Also other hybrid methods are presented. For example some of them use caching in order to improving performance [10].

Hardware approaches typically use dedicated hardware for routing lookup [5, 11]. More popular techniques use commercially available content-addressable memory (CAM). CAM storage architectures became popular because their searching time is *O* (1) that is; it is bounded by a single memory access. Binary CAMs allow only fixed-length comparisons and so they are not suitable for longest-prefix matching. The TCAM solves the longest-prefix problem and is by far the fastest hardware device for routing. Unlike to TCAMs, ASICs that use trie— digital trees in storing strings (in this case, the prefixes) [12]—require 4 to 6 memory accesses for a single route lookup and thus causes longer latencies. Also, TCAM-based routing table updates have been faster than their trie based counterparts. The number of routing-table entries is increasing super linearly [12]. Today, routing tables have approximately 500,000 entries [13], also it is very important to attention the need of optimal storage. Yet CAM vendors claim to handle a maximum of only 8,000 to 128,000 prefixes, taking allocators and deal locators into account [12].The gap between the projected numbers of routing-table entries and the low capacity of commercial products has given rise to work on optimizing the TCAM storage space by using the properties of lookup tables [14, 15]. Even though TCAMs can store large numbers of prefixes, they consume large amounts of power, which limits their usefulness. Panigrahy and Sharma introduced a paged-TCAM architecture to reduce power consumption in TCAM routers[16].Their scheme partitions prefixes into eight groups of equal size; each resides on a separate TCAM chip. A lookup operation can then select and enable only one of the eight chips to find a match for an incoming IP address. In addition, the approach introduces a paging scheme to enable only a set of pages within a TCAM. However, this approach achieves only marginal power savings at the cost of additional memory and lookup delay. Other work describes two architectures, bit selection and trie-based, which use a paging scheme as the basis for a power-efficient TCAM [17]. The bit selection scheme extracts the 16 most significant bits of the IP address and uses a hash function to enable the lookup of a page in the TCAM chip. The approach assumes the prefix length to be from 16 to 24 bits. Prefixes outside this range receive special handling; the lookup searches for them separately. However, the number of such prefixes in today routers is very large (more than 65,068 for the bbnplanet router) [13], and so this approach will result in significant power consumption. In addition, the partitioning scheme creates a trie structure for the routing table prefixes and then traverses the trie to create partitions by grouping prefixes having the same sub prefix. The sub prefixes go into an index TCAM, which further indexes into static RAM to identify and enable the page in the data TCAM that stores the prefixes. The index TCAM is quite large for smaller page sizes and is a key factor in power consumption. The three-level architecture, though pipelined, introduces considerable delay. It is important to note that both the approaches [16, 17] store the entire routing table, which is unnecessary overhead in terms of memory and power. The existing approaches reduce power either by routing table compaction or by selecting a portion of the TCAM. The proposed approach in this paper is based on TCAM partitioning but its innovation is selecting optimum part to start IP lookup operation. Notice that selecting this part has no significant overhead on system. In the next section this problem will be discussed.

3. EASTFILE Architecture

Initially, the architecture of the proposed approach and its phases will be discussed. Notice, it is supposed that the lookup table minimization has been done with common techniques such as Espresso-II and we don't explain this matter anymore. You can get more information about these techniques in [18]. Figure 1 presents the block diagram of this architecture. The operation of this system is surveying in four following phases:

- Phase 1 is entitled ISU that is the abbreviation for IP Splitter Unit.
- Phase 2 is related to finding the rank of each part of IP address which has split in the previous phase. The result of this ranking will be stored in RMMs or Ranking Memory Modules. We will discuss more about the ranking later.
- Phase 3 is called MDU, that is stand for Minimum Detector Unit.
- Phase 4 is entitled LSU or Longest Prefix Match Selector Unit.

At first suppose that the addresses are only IPv4 and then the scalability of this opinion for IPv6 will be presented. Now, each phase will be explained in details separately.

3.1. Phase 1: ISU (IP Splitter Unit)

This phase doesn't have so much complexity. The operation of ISU is just splitting IP addresses and storing these parts in data registers which are illustrated in figure 1. as DR1 to DR4. Since the addresses are 32 bits in IPv4, each of the DRs will have 8 bits. Certainly dividing the 32 bits data to four parts will be provided by very simple circuits. So it will be found that the first phase is very simple and without any unusual overhead.

3.2. Phase 2: RMM (Ranking Memory Modules)

The most important innovation in our approach is occurred in this phase. Indeed this phase helps us to control the amount of power consumption. Partitioning the IP lookup tables is common and is presented in some IP lookup techniques [19-21]. All of which, start looking up the table from the 1st level to the last one. Figure 2 illustrates the overview of those techniques. Whereas the manufacturers of CAM believe that their products are suitable for tables with at most 128000 entities [13] although the memory accesses will increase by partitioning, using TCAM will be feasible for large IP lookup tables by this method. Obviously, the power consumption in TCAMs depends on the enabled bits of lookup process. Partitioning the IP lookup table is useful to decrease the number of enabled bits in the lookup process. But why looking up the table should be started from the 1st level? Could it be started from the second or third level? Is the initial selected level important? This proposed approach can answer to all of these questions. The answer of the first question is No. We can start looking up the table from each part, but the lookup must be circular. The answer of the last question is very important, too. Since the partitioned lookup operation sieves the table rows, the best part for the first lookup is a level which has the most number of sieved rows. In other words, the part which has the least number of rows that are equal to the search key is the best choice for 1st level of lookup. Thus, if the solution to predict the matching rate of each part can be found, then search can be started from the part that has the lowest match rate. Because this part has the most screening so fewer rows of the next part will be active. This technique helps the uninformed search not to be done.

Since each part in IPv4 contained 8 bits, so the range of numbers can be 0 to 255. Therefore frequency rate of these numbers in each part should be stored in its related RMM. So a RMM has 256 rows that each row stores the frequency rate of the corresponding number row in the relevant part. The frequency rate of a number in each part is called rank of that number. For example if 150 prefixes start with 192 in the lookup table, the rank of 192 in the first RMM will be 150 or if 200 prefixes have 0 number in their second part so the rank of 0 in the second RMM will be 200. That is, the 193th row of first RMM stores 150 and the 1th row of second RMM stores 200. Notice, the 1th row of each RMM stores the rank of 0 and 256th row is related to the rank of 255. So the Rank_j (i) will be defined as follow:

$$Rank_j(i) = the\ frequency\ rate\ of\ \ i\ in\ part\ j\ (j^{th}\ TCAM)$$

If each row of RMMs has 24 bits, the biggest possible rank will be 2^{24}. When the rank of n in a part is equal to total IP lookup table rows, the worst case will be occurred. In this case the rank of a number has its maximum value. It's possible to have IP lookup tables with 2^{24} (16 millions) rows in this case. This capacity of IP lookup tables is so more than today and future needs. Equation (1) and (2) shows the capacity of each RMM and the total capacity of RMMs, respectively.

$$256 \times 24\ bits = 6144\ bits = 6\ Kbit \quad (1)$$

$$4 \times 6\ Kb = 24\ Kb = 3\ KByte \quad (2)$$

It's possible to implement RMMs with SRAM. Also in each IP lookup, only 96 (4×24) bits are activated which doesn't violate the power consumption optimizing. The way of filling the RMMs explained later.

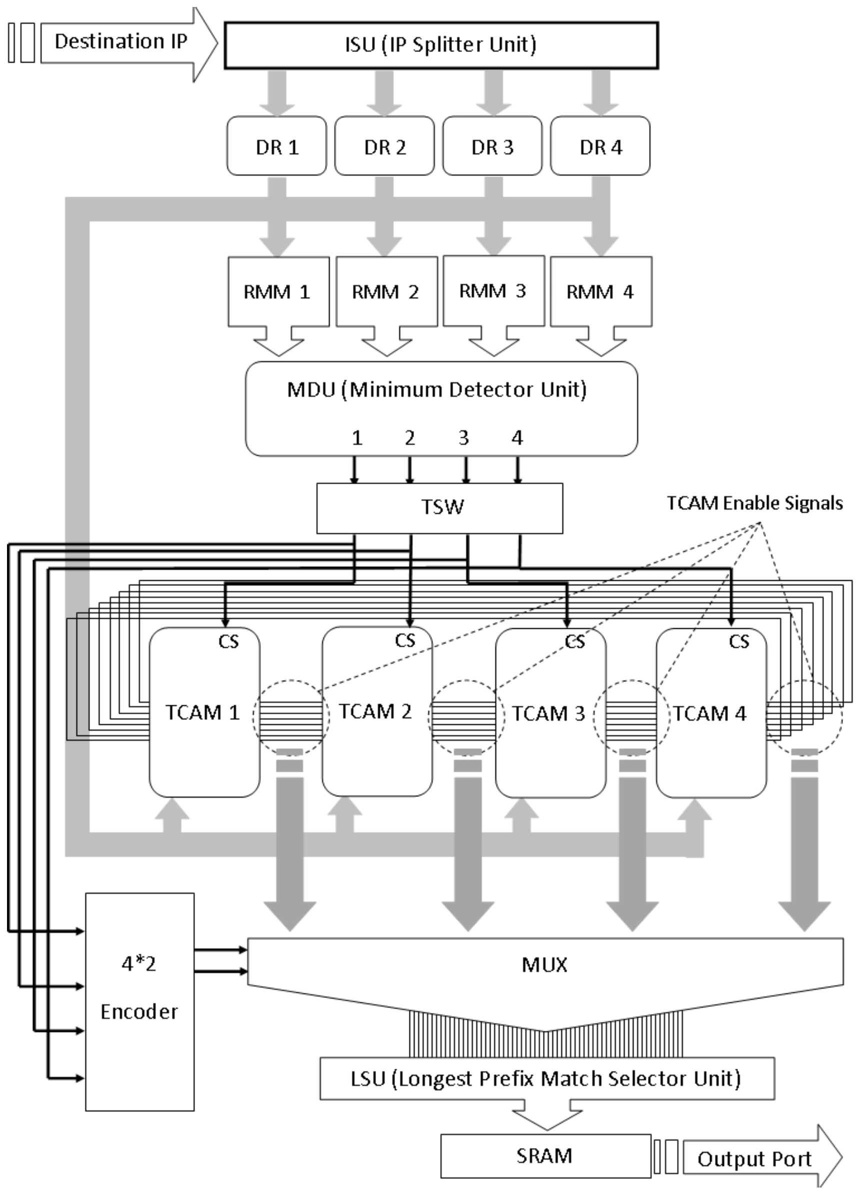

Figure 1. The block diagram of EASTFILE architecture.

3.3. Phase 3: MDU (Minimum Detector Unit)

In the previous phase, the rank of numbers in each part was determined. Now, selection of the best part for starting lookup is the main problem. The others multi level enabling techniques start lookup from the first level, but it's not the best choice. The best candidate for starting lookup is a part which has the minimum matches with the search key. This part is that which has the minimum rank of search key. Notice, in the first step, all rows of the starting level must be active; because at this time we have no idea for selection. Suppose that the total number of prefixes equal to n and e_i indicates the proportion of match rows count in i^{th} step of lookup to n. Equation (3) shows the total of enabled bit in lookup process.

$$8n + 8ne_1 + 8ne_2 + 8ne_3 = 8n(1 + e_1 + e_2 + e_3) \quad 0 \le e_i \le 1 \tag{3}$$

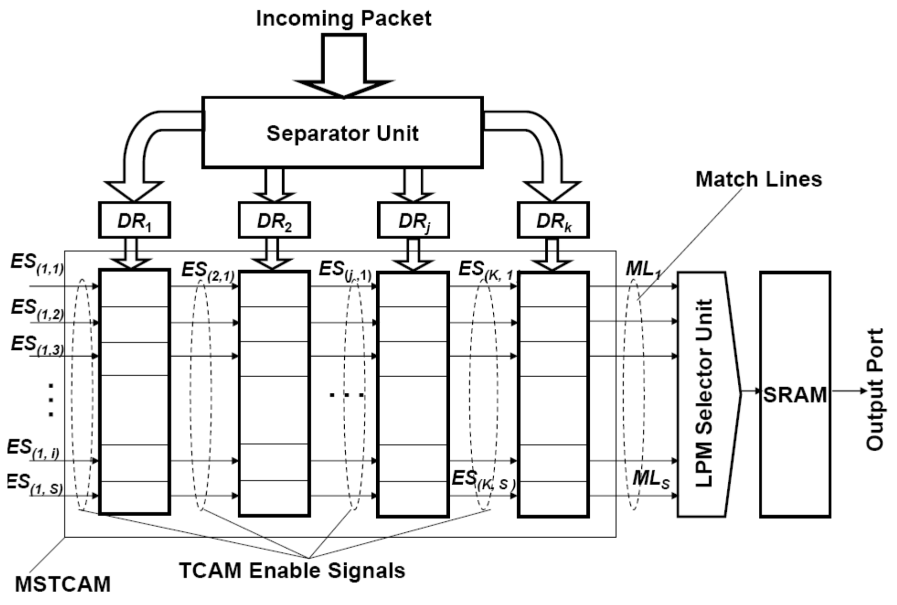

Figure 2. Block diagram of TCAM partitioning techniques.

The range of e_i has been selected between 0 and 1, because in the minimum case the count of matches equal to 0 and in the maximum case equal to n. Note that values of e_i are non-ascending. Because in lookup process in each step some number of rows are set aside. This is the screening feature that described before. Equation (4) represents this fact.

$$e_3 \le e_2 \le e_1 \le n \tag{4}$$

Therefore MDU is a simple 24 bits minimum detector. For example if table 1 shows the value of RMMs, for IP address 69.144.56.0, we start lookup from 4th part. In fact MDU takes the minimum between 59753, 3410, 246 and 1 (these numbers has gray color in the table). In this selection, only one row of TCAMs is enabled except in the first part where we have to active all of its rows.

Table 1. Instance of RMM.

Numbers	RMM 1	RMM 2	RMM 3	RMM 4
0	59832	4082	236	1
…	…	…	…	…
56	59726	2680	246	10
…	…	…	…	…
69	59753	3229	165	4521
…	…	…	…	….
144	59764	3410	273	0
…	…	…	…	…

The result of MDU is a control code that is stored in TSW (TCAM Selection Word). In the previous example, TSW equals to 0001. Considering figure 1 each bit of TSW is connected to relevant TCAM chip select input. Thus in this example TCAM 4 is selected in the first step and TSW must be shifting right circularly three times in the next steps.

3.4. Phase 4: LSU(LPM Selector Unit)

As the last stage of lookup process, longest prefix is selected from active lines in TCAMs. Indeed this phase explains how the longest prefix matching is done. Since the last part of lookup can be variant, so the match lines of all TCAMs are connected to a multiplexer as shown in figure 1. The select inputs of this multiplexer are provide by an encoder. The inputs of encoder are connected to TSW. For example if 0100 is supposed as the last data in TSW, then the encoder outputs will be 10 and the third input of multiplexer is selected. That is, the last stage has been the third TCAM. Now, considering the matched lines, the longest prefix will be selected. If the prefixes are ordered descending in the lookup table, then it will be possible to use a priority encoder for selecting the longest prefix [19-21].

4. Performance evaluation

In this section performance evolution is presented. This section will be discussed in three parts as follows: (1) the cost of main operations surveying, (2) the power consumption performance and (3) the scalability.

4.1. The cost of main operations

Each IP lookup engine should do lookup, insert, delete and update operations. Although lookup is the most important operation; but if an IP lookup engine can't be able to do other operations efficiently, this IP lookup engine won't be acceptable. The surveying of each operation will be presented separately.

4.1.1. Lookup operation

When a new packet is received at one of the input port of router, the ISU extracts the destination IP address from the IP header and prepares it for each stage by split the address and store split parts in DRs. Lookup begins with activation of RMMs which specify the rank of each number that has been stored in DRs. Then obtained ranks are compared with each other in MDU. The result of MDU is a control code that is stored in TSW which presents the starting stage for lookup. The next step is a four stage lookup in TCAMs. In this step the match lines for each TCAM provides the enable lines of the next TCAM, circularly. If TSW indicates the i^{th} TCAM for starting lookup, then the i^{th} stage is enabled and 8 bits of most significant bits of IP address which are stored in DR_i are compared with all rows in $stage_i$ simultaneously and rest of the work will be continued as mentioned before. Finally LSU determines the longest prefix which has been matched with destination IP address. Figure 3 shows the activity diagram of this operation. The main activities of this operation are summarized as follow:

- IP address splitting
- Finding rank of parts
- Minimum rank detection
- TCAMs lookup
- Longest prefix match detection

Clearly, splitting IP address is very simple and doesn't have any obvious overhead. Since the rank of n is the value of $(n+1)^{th}$ row of RMM, finding the rank of parts is done simply too. For example the rank of 192 has been stored in 193th row of RMM or the rank of 0 is existed in the first row of RMM. Note, finding the rank of all parts is done simultaneously. Detection of the minimum rank is done by a simple minimum detection circuit and isn't complex. The most overhead belongs to searching TCAMs. This step needs four memory (TCAM) accesses. Finally, as mentioned before in section 3.4, longest prefix match selector unit isn't complex. Therefore, lookup operation in the proposed approach is O(1) which is so fast.

4.1.2. Insert operation

The only suspicious point in the insert process is changing the rank of numbers. In this task, the increasing signal of relevant rows of RMMs is enabled. For example if we want to insert the 192.168.0.1 into the table, then 193th row of RMM1, 169th row of RMM2, 1th row of RMM3 and 2th row of RMM4 should be increased. When prefixes have been inserted instead of IP addresses, this problem will be more challenging. In these cases the ranks of many numbers increase in each part which has don't care bits. For example if the prefix is equal to 4.23.112.0/24, then $Rank_1(4)$, $Rank_2(23)$, $Rank_3(112)$ and the rank of all numbers in 4th part will be increased; or if we want to insert the 4.23.112.0/23 into the table, then the 3th and 4th parts will have don't care bits. In this case $Rank_1(4)$, $Rank_2(23)$ and rank of all numbers in 4th part will be increased. But in the 3th part, the pattern of increasing is equal to 0111000* i.e. the rank of 01110000 (112) and the rank of 01110001 (113) should be increased. Notice, 4.23.112.0/23 is equal to 00000100 00010111 0111000* ********, if * indicates don't care bits. So, as mentioned before the rank of all numbers in 4th part and the rank of 112 (01110000) and 113 (01110001) in 3th part should be increased. Therefore if we can increase the value of RMM rows simultaneously, then the insert operation won't be complex and it is O(1).

4.1.3. Withdrawal

The algorithm for removing prefixes from the table isn't complex. This operation is the same as insert, but when an IP have been removed, the rank of some numbers must be decreased. For example, removing 4.23.112.0 leads to decreasing of $Rank_1(4)$, $Rank_2(23)$, $Rank_3(112)$ and $Rank_4(0)$. Indeed if $Rank_1(4)$ is equal to x, after deletion of that IP, $Rank_1(4)$ will be equaled to x-1. When prefixes have been deleted instead of IP addresses, ranks of many numbers decrease in each part which has don't care bits. For example, if we want to delete the 4.23.112.0/23 from the table, then the 3th and 4th parts will have don't care bits. In this case $Rank_1(4)$, $Rank_2(23)$ and rank of all numbers in 4th part will be decreased. But in the 3th part, the pattern of decreasing is equal to 0111000* i.e. the rank of 01110000 (112) and the rank of 01110001 (113) should be decreased. Therefore if we able to decrease the value of RMM rows simultaneously, then the withdrawal operation will be O(1) and it isn't so costly.

4.1.4. Update operation

Approximately 100 to 1,000 updates per second take place in core routers today [22].Thus, the update operation should be incremental and fast to avoid becoming a bottleneck in the search operation. Update operation include two sub operations: insert and withdrawal. Therefore the update operation isn't complicated; because as mentioned before, the insert and the withdrawal operations are simple.

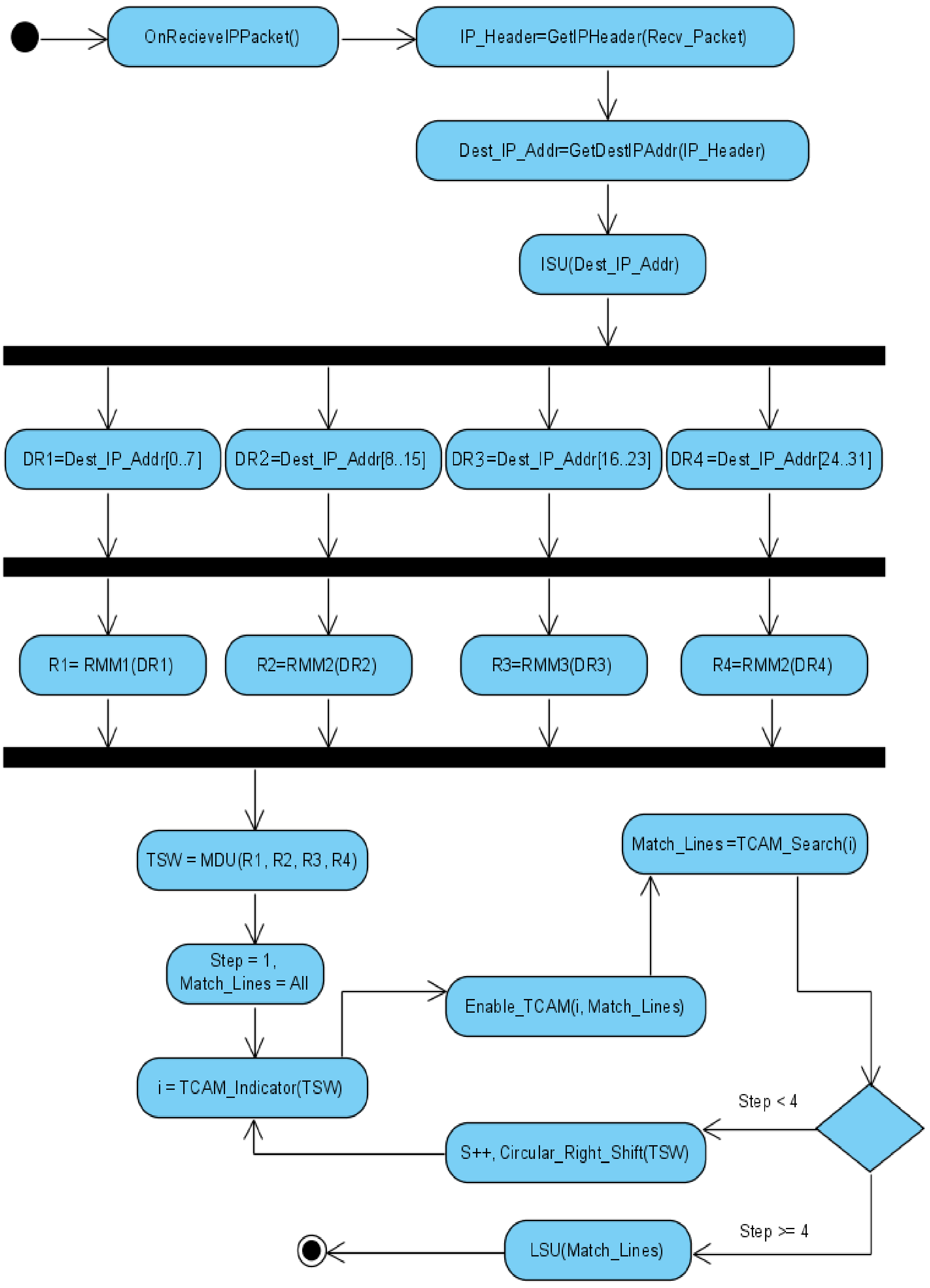

Figure 3. Lookup activity diagram.

4.2. Power consumption performance

The most important feature of EASTFILE is power consumption efficiency. 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 0s and 1s. Since the contents of a TCAM can be searched in parallel and the first matching result can be returned within only a single memory access, TCAM-based scheme is very promising in terms of building a high-speed LMP lookup engine [6]. Moreover, TCAM-based tables are typically much easier to manage and update than trie-based ones [6]. Although TCAMs are very cost effective and simple to manage, their power consumption is very high and this feature is the mainspring of their failure. The high power consumption is in contrast with scalability for TCAM based IP lookup engines. As mentioned before the RMMs can be implemented with SRAM and their total capacity is just 3 Kbyte. Therefore the RMMs aren't able to play an important effect on power consumption. The other components in EASTFILE except TCAMs don't have high power consumption too.

4.2.1. The referenced model

If the IP lookup table isn't partitioned, then just one TCAM level will be existed. This architecture is called referenced model. Figure 4 shows the block diagram of referenced model architecture.

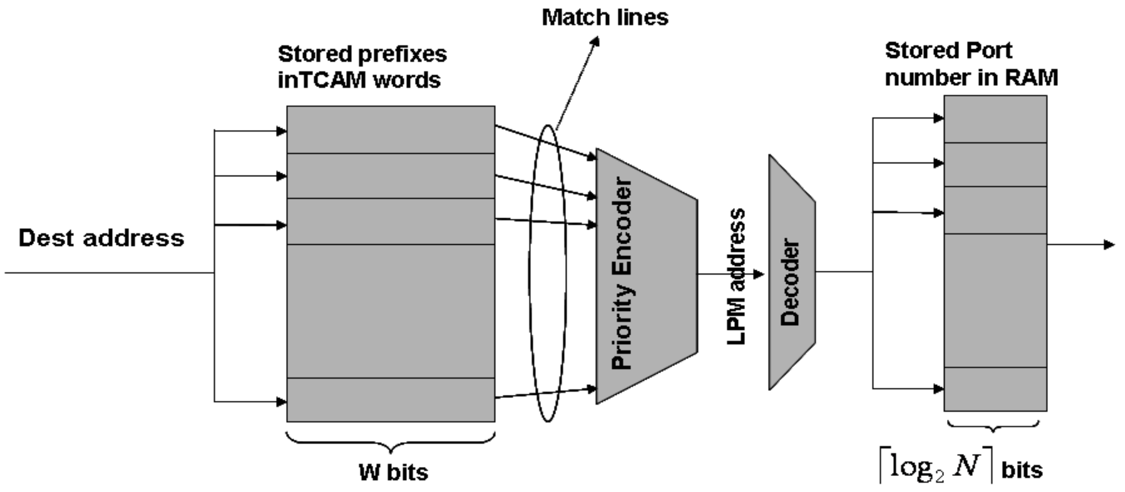
Match lines
Stored prefixes inTCAM words
Stored Port number in RAM
Dest address
Priority Encoder
LPM address
Decoder
W bits
$\lceil \log_2 N \rceil$ bits

Figure 4. The block diagram of referenced model.

Referenced model presents the simplest TCAM based IP lookup engine. This architecture isn't scalable for large IP lookup tables and IPv6; because in this model the lookup operation leads to enabling all TCAM row bits. However this model lookups the destination IP address with one memory access; but it has the highest power consumption rather than all other TCAM based IP lookup engines.

4.2.2. EBPS (Enabled Bits Per Search)

The EBPS is defined as the number of enabled bits in each search. It is the main parameter in power consumption measurement and evaluation. If r represents the count of enabled rows and w shows the number of bits in each row, then equation (5) illustrates the EBPS.

$$EBPS = r \times w \qquad (5)$$

If the IP lookup table has n rows, Equation (6) will show the EBPS in the referenced model.

$$EBPS_{Ref} = n \times w \qquad (6)$$

In equation (6) w shows the number of bits for IP addresses which is equal to 32 bits for IPv4 and 128 bits in IPv6 tables. Since in the referenced model, all TCAM row bits are enabled in each lookup, the maximum of EBPS is equal to $EBPS_{Ref}$.

$$EBPS_{Max} = EBPS_{Ref} = n \times w \quad (w = 32 \; or \; 128) \qquad (7)$$

In the proposed architecture for EASTFILE, EBPS can be computed as equation (8). Assume that the total number of prefixes represents by n and e_i indicates the proportion of match rows count in i^th step of lookup to n.

$$EBPS = 8n + 8ne_1 + 8ne_2 + 8ne_3 = 8n(1 + e_1 + e_2 + e_3) \quad 0 \le e_i \le 1 \qquad (8)$$

Considering equation (4), in the best case, e_1 must have the minimum value. Since the part that has the lowest rank is selected by MDU, so the EPBS in the proposed approach will be minimal. If R_i is equal to the Maximum rank in j^th TCAM, then equation (10) shows the maximum of e_i.

$$R_i = \text{Max}(Rank_i(j)) \qquad 1 \le i \le 4,\; 0 \le j \le 255 \qquad (9)$$

$$M_i = Max(e_i) = \frac{R_i}{n} \quad 1 \le i \le 4 \qquad (10)$$

In the worst case, the all parts of destination IP address have maximum rank in their level. For example, if IP address is equal to x.y.z.w and $Rank_1(x) = R_1$, $Rank_2(y) = R_2$, $Rank_3(z) = R_3$ and $Rank_4(w) = R_4$ then the worst case will be occurred. In this case, MDU select the minimum of $\{R_1, R_2, R_3, R_4\}$ and lookup start from the part which has the minimum rank. .Therefore if ω indicates the power consumption of one TCAM cell and P_{Max} shows the maximum of power consumption in proposed approach, then the equation (11) illustrate the P_{Max}.

$$P_{Max} = 8n\omega \times \left(1 + \frac{\min(R_1, R_2, R_3, R_4)}{n} + e_2 + e_3\right) = 8n\omega \times (1 + \min(M_i) + e_2 + e_3) \qquad (11)$$

Considering equation (4), we know the value of e_3 and e_2 are less than e_1 so equation (11) can be rewrite as follows:

$$P_{Max} \le 8n\omega(1 + 3 \times \min(M_i)) \qquad (12)$$

Therefore equation (12) shows that the power consumption in EASTFILE is bounded. We will discuss more about this topic in the experimental results section.

4.2.3. POF(Power Optimization Factor)

This parameter refers to the minimum power optimization percentage which is defined as equation (13).

$$POF = \frac{P-P_{Max}}{P} \times 100 \qquad (13)$$

In equation (13), P is defined as power consumption in arbitrary TCAM based IP lookup engine.For example, POF_{Ref} is shown in equation (14).

$$POF_{Ref} = \frac{P_{Ref}-P_{Max}}{P_{Ref}} \times 100 = \frac{EBPS_{Ref} \times \omega - 8n\omega \times (1+\min(M_i)+e_2+e_3)}{EBPS_{Ref} \times \omega} \times 100 = \frac{nw\omega - 8n\omega(1+\min(M_i)+e_2+e_3)}{nw\omega} \times 100 = \frac{n\omega(w-8(1+\min(M_i)+e_2+e_3))}{nw\omega} \times 100 = \frac{w-8(1+\min(M_i)+e_2+e_3)}{w} \times 100$$
(14)

Suppose that the all addresses in table are IPv4 addresses, so w is equal to 32 and equation (14) can be rewrite as follows:

$$POF_{Ref} = \frac{32-8(1+\min(M_i)+e_2+e_3)}{32} \times 100 \qquad (15)$$

As noted before the upper bound of P_{Max} is shown in equation (12). Considering this bound and equation (15) we able to define a lower bound for POF_{Ref}. If we replace the P_{Max} with its upper bound, then equation (15) will be modified as equation (16).

$$POF_{Ref} = \frac{32-8(1+3\min(M_i))}{32} \times 100 = \frac{4-1-3\min(M_i)}{4} \times 100 = \frac{3}{4}(1-\min(M_i)) \times 100 \qquad (16)$$

As noted before, we will discuss more about performance of power consumption in experimental results section.

4.3. The Scalability

As noted before, a good IP lookup scheme should have a solution for scalability. Scalability refers to the performance of the IP lookup scheme which neither depend on the number of prefixes in lookup table nor the number of IP address bits. That is, the IP lookup engine must be effective for large IPv4 or IPv6 lookup tables. In sections 4.1 and 4.2 it was shown that the proposed approach is scalable for large IPv4 lookup tables; because all of the main operations in this scheme are effective. Also the power consumption in this approach is very cost effective. Therefore if this approach is scalable for IPv6 addresses, then it will be acceptable as an appropriate scheme.

4.3.1. The IPv6 Address allocation policy

Figure 5 illustrates the address allocation policy in IPv6, which is administrated by IANA to regional registries: APNIC, ARIN and PIPE. The global unicast address is partitioned into several segments as a hierarchical tree such as ISP, Site or LAN [23].

The format of the global unicast IPv6 addresses is shown in figure 6.

The interface identifier portion of an IPv6 address can be configured either manually or automatically. The interface identifier must be unique within the link to which the interface is attached regardless of the configuration method.[RFC3513] requires that all unicast addresses except those starting with binary bits 000 be constructed using the modified IEEE EUI-64 format. We first describe the original EUI-64 identifier format as in Figure 7. The EUI-64 identifier contains a 24-bit Organizationally Unique Identifier (OUI), or company-id, and a 40-bit extension identifier. Within the company-id the universal/local bit indicates whether the 64-bit identifier is globally administered or locally administered (this bit is set if it is local). The group/individual bit indicates whether the identifier identifies a single hardware instance or a group of hardware instances (this bit is set for a group). The IEEE MAC-48 hardware interface address has a similar format except MAC-48 has a 24-bit extension identifier. Using EUI-64 format the MAC-48 hardware interface address can be converted into a 64-bit interface identifier when IPv6 operates over IEEE 802-based networks. The conversion method inserts 0xFFFE into the MAC-48 address expanding it to 64 bits, and then inverts the universal/local bit. For example, if a MAC-48 address is 00-60-97-8F-6A-4E, then after conversion the final interface identifier is 02-60-97-FF-FE-8F-6A-4E as shown in Figure 8 [24].

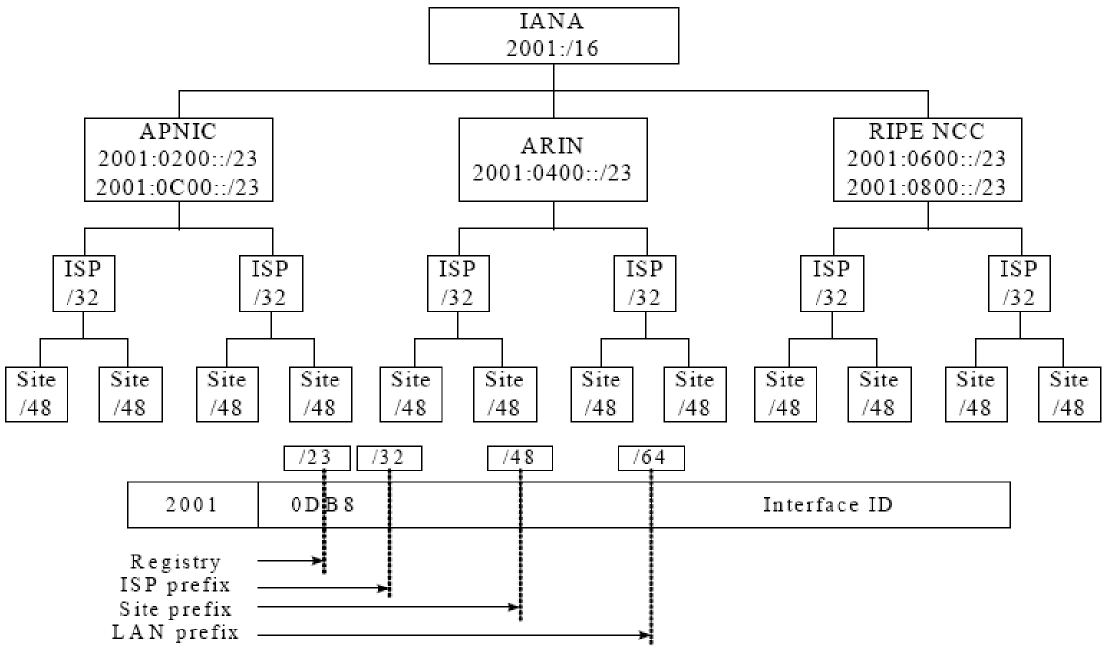

Figure 5. Address allocation policy.

3 bits	13 bits	8 bits	24 bits	16 bits	64 bits
FP	TLA ID	RES	TLA ID	SLA ID	Interface ID

Figure 6. The format of global unicast IPv6 addresses.

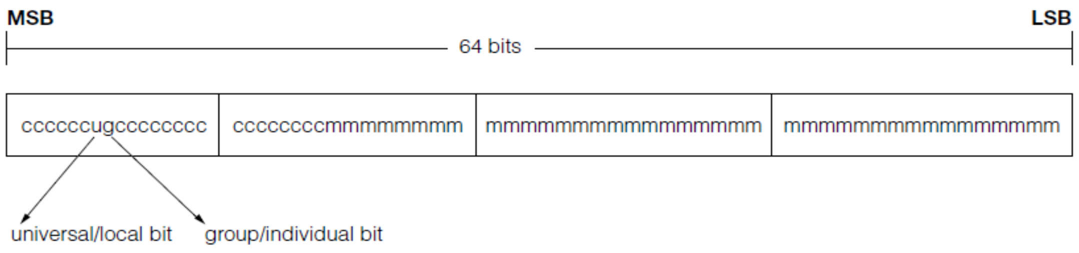

Figure 7. IEEE EUI-64 identifier format.

The “modified” format is different from the IEEE’s standard extension from a MAC-48 address to the EUI-64 identifier in the following two points:

- The universal/local bit is inverted.
- The inserted 16-bit value is 0xFFFE while 0xFFFF should be used for the standard extension.

As the result of this section, IPv6 prefixes are at most 64 bit and this is very important in the Internet routers design.

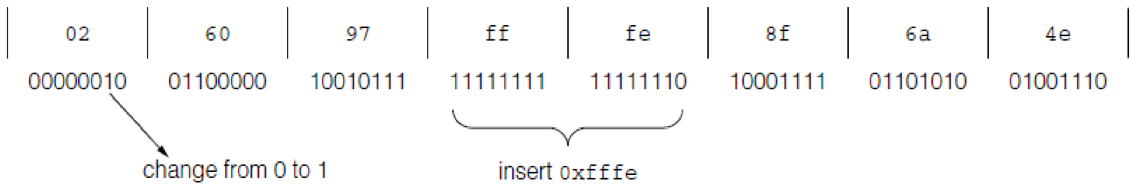

Figure 8. MAC-48 to 64-bit interface identifier conversion.

4.3.2. Summary of the IPv6 scalability

Considering the IPv6 address allocation policy and the length of the IPv6 prefixes, it's enough to divide the IPv6 addresses into 16, 16, 16, 16, 64 bit in the proposed approach scalability. Note that the last part of this partitioning isn't useful in the lookup operation unless the IPv6 addresses instead of the IPv6 prefixes are stored in the IP lookup table; albeit this fact occurs very rarely. Therefore, only the 16 bit parts will take apart in the ranking. Now, it should be mentioned that the proposed approach will be tolerable after this change. The only part of this architecture which seems to be suspicious is the ranking memory modules. Note that the ISU is easy to adopt for the IPv6 partitioning and the operation of the MDU and the LSU are not different from the IPv4.

4.3.2.1. RMMs in IPv6

Since each part of the IPv6 addresses are 16 bit, so the counts of the numbers which are in a part will be 2^{16}. Therefore, each RMM should have 2^{16} rows. Also, considering 24 bits for each RMM row is enough in the IPv6 lookup tables, too. Equation (17) and equation (18) illustrate the capacity of each RMM and total capacity for all RMMs.

$$2^{16} \times 24\,bit = 1536\,Kbit = 192\,KByte \qquad (17)$$

$$4 \times 192\,KByte = 768\,KByte \qquad (18)$$

The received capacity in the above equation shows that it's feasible to implement the RMMs in this scheme, because the total capacity of RMMs is less than 1MByte.

4.3.2.2. Performance evaluation in IPv6

In the proposed IPv6 method, the insert and the withdrawal operations are not different with IPv4. But the lookup operation in IPv6 requires one more memory access rather than IPv4, this increment is related to the last 64 bit part. When IPv6 is stored in the IP lookup table instead of the IPv6 prefix, this access will be required.

The power consumption equations in IPv6 will be modified as follows:

$$EBPS = 16n + 16ne_1 + 16ne_2 + 16ne_3 = 16n(1 + e_1 + e_2 + e_3) \quad 0 \le e_i \le 1 \qquad (19)$$

$$P_{Max} = 16n\omega \times \left(1 + \frac{\min(R_1,R_2,R_3,R_4)}{n} + e_2 + e_3\right) = 16n\omega \times (1 + \min(M_i) + e_2 + e_3) \quad (20)$$

$$P_{Max} \leq 16n\omega(1 + 3 \times \min(M_i)) \quad (21)$$

5. Experimental Results

In this section we present some exciting experimental results to prove the correctness of our proposed idea. We applied and analyzed real data, which is downloaded from [25] and process it in the new software which is called IP lookup analyzer. In the next subsection we explain more about this software.

5.1. IP lookup analyzer

IP lookup analyzer is a simple software program which is programmed by our team and is very useful to obtain some experimental results. Activities of this program summarize as follows:

- Processing the received raw data (Internet routers report files)
- Conversion of raw data to a relational table
- Extraction of useful data from the created table
- Calculation the rank of the numbers via generating some unproblematic SQL queries
- Generating random IP addresses and creating statistical data

Thus IP lookup analyzer helps to evaluate correctness of considering idea on real IP lookup data.

5.2. Test bed: a real BGP table

In this subsection we present the specifications of a real BGP table which are obtained from AS65000 APNIC R&D report [25]. This report has been updated at Sun Aug 22 2010. Figure 9 shows the growth of the Active BGP entries (FIB) in this router from 1989 to present and table size metrics are presented in table 2.

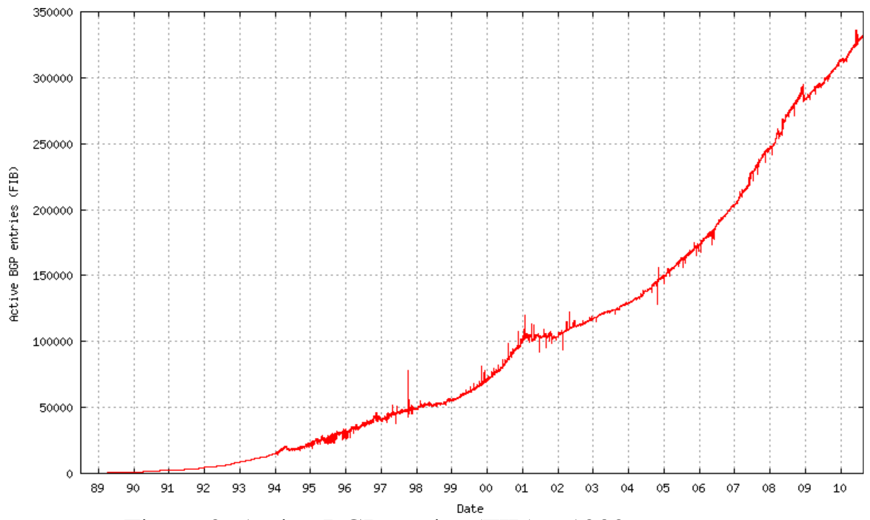

Figure 9. Active BGP entries (FIB) – 1989 to present

Table 2 indicates, this router report is an appropriate benchmark to prove the correctness of our idea because the IP lookup table of this router is large enough. So the BGP table text file in this report will be selected as a raw data for the IP lookup analyzer.

Table 2. Table size metrics

FIB / RIB Table Reports	Data Sets
Active BGP entries (FIB)	333513
All BGP entries (RIB)	658483
RIB/FIB ratio (658483/333513)	1.9744
Valid entries	658483

5.3. Results

The IP lookup analyzer output and its statistical results have been presented in this subsection. The ranking result in each part of the IP lookup table and summery of this ranking are shown in figure 10 and table 3, respectively.

Table 3. Summery of ranking result

Size of table	333513			
Parts(bit index)	[0..7]	[8..15]	[16..23]	[24..31]
IP	205	16	0	0
R_i	12612	2586	26229	331721
min(R_i)	2586			
Mi=R_i/n	0.037816	0.007754	0.078645	0.994627
min(M_i)	0.007753821			

Considering the table 3 and the received equation for power consumption, equation (22) shows the P_{max} in this case.

$$P_{Max} \leq 8n\omega(1 + 3 \times 0.007753821) \approx 8n\omega \times 1.023 \quad (22)$$

Also equation (23) illustrates the POF_{ref} as follows.

$$POF_{Ref} = \frac{3}{4}(1 - 0.007753821) \times 100 \approx 74.42\% \quad (23)$$

Equation (23) indicates that in the worst case, EASTFILE can save 74% of routing table's power consumption in comparison with the reference model. To compare EASTFILE with the simple partitioning technique, 100000 IP address via the IP lookup analyzer have been generated and we obtained some interesting results shown in table 4.

Table 4: EASTFILE vs. the simple partitioning technique

1	Total IP addresses	100000
2	First part selected number (as same as the simple partitioning technique)	75530
3	Other parts selected number (against of the simple partitioning technique)	24470
4	Maximum difference of the match lines number	11502
5	Average difference of the match lines number	599.13
6	Average match lines number of the simple partitioning technique	1371.46
7	Average match lines number of EASTFILE	772.33

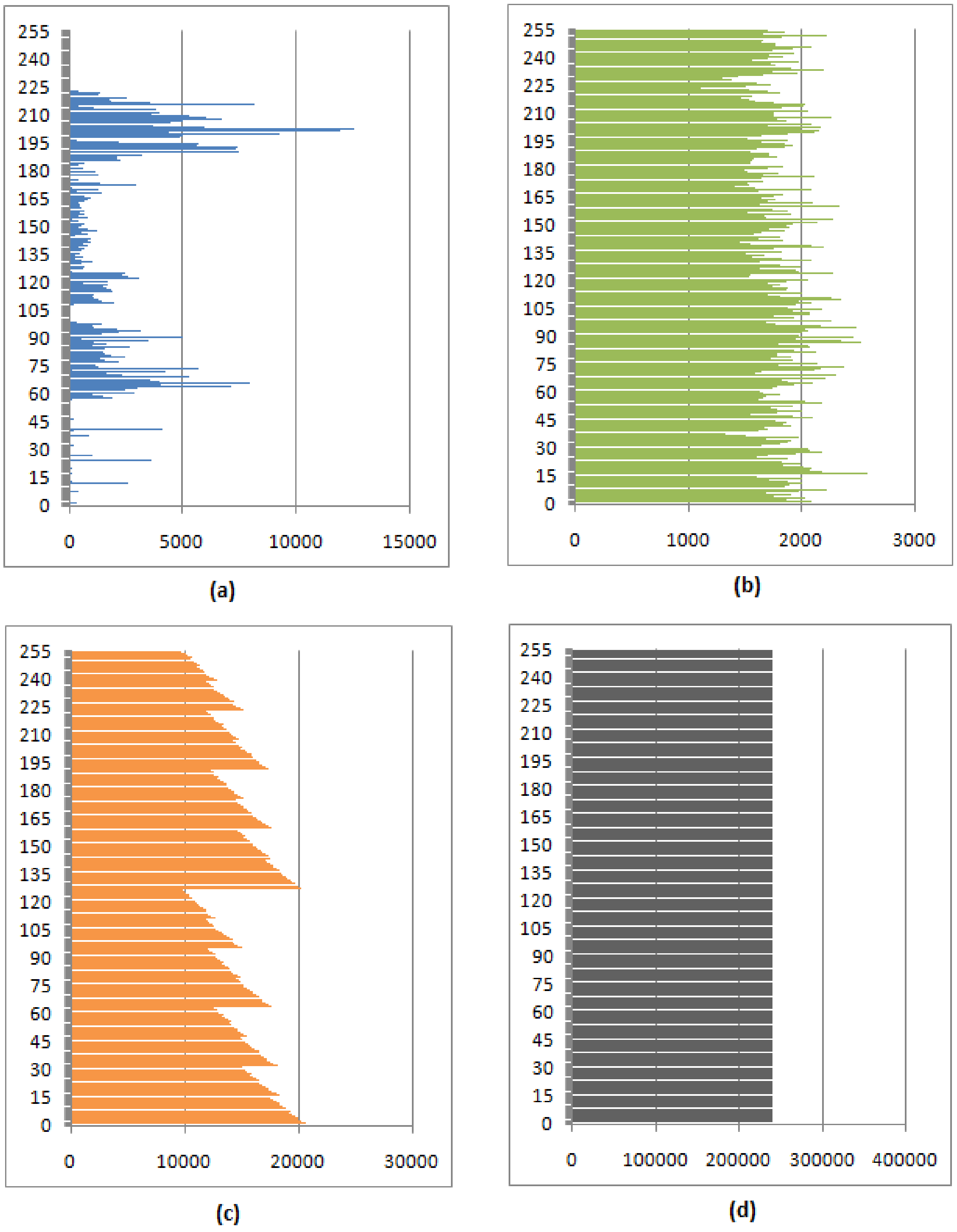

Figure 10. The ranking result in the IP lookup table. (a), (b), (c) and (d) show the ranking result in parts 1 to 4, respectively.

Table 4 indicates the following points:

- The number of IP Addresses which are examined by IP lookup analyzer is shown in the first row.
- The second row shows that in 76 percent of lookups, both methods must be started from the first part, in other word in these cases the first part is the best choice for starting lookup operation.
- The third row indicates EASTFILE is more intelligent than other technique in 24 percent of lookups because, it selects the best part instead of the first one for starting lookup operation.
- The forth row specifies that the maximum difference between these two methods for math lines number is 11502, in the first step of lookup operation. This difference can occur when 203.224.127.165 is looked up. Because the rank of 203 is 12612 but the minimum rank of these numbers is 1110. Indeed EASTFILE starts the lookup operation for this IP from the second part.
- The average difference of the match lines number between these two methods is 599.13 which is shown in fifth row.
- Sixth and seventh rows of table indicate that, in the first step of lookup operation in the simple partitioning technique 1371.46 and in EASTFILE 772.33 match lines number occurs by average. Consequently EASTFILE has 46% match lines number less than the simple partitioning technique by average.

6. Conclusion and future work

The significantly increased IP lookup table rows and the address length of IPv6, pose a great challenge on wire-speed packet forwarding in high-end routing devices. In this paper, we propose a novel and intelligent IP lookup engine that based on TCAM. Our scheme is scalable for large IP lookup table and IPv6. We also develop an architecture named EASTFILE which is energy aware scalable TCAM based fast IP lookup engine. This architecture is very useful not only because of the decrease power consumption but also because of the power consumption bounding. This architecture needs the 5 memory accesses for IPv4 lookup and 6 memory accesses for IPv6 lookup in the worst, average and the best case.

The experimental result shows that this approach decreases the power consumption approximately 74% rather than the referenced architecture and also decreases 46% number of match lines in the first step of lookup operation rather than the other multi level enabling techniques or simple partitioning technique. Therefore this architecture can be so fast, cost and power effective.

The ongoing research of our team will cover: (1) further improving the power consumption with dynamic partitioning lookup table; (2) extending the propose scheme to support applications that require multi field searching such as packet classification; (3) improving the prefix generator algorithms such as Espresso-II.

REFERENCES

[1]. D.R. Morrison, PATRICIA – practical algorithm to retrieve information coded in alphanumeric, J. ACM 15 (4) (1968) 514–534.

[2]. S. Nilsson, G. Karlsson, IP-address lookup using LC-tries, IEEE J. Select. Areas Commun. 17 (6) (1999).

[3]. A. Donnelly, T. Deegan, IP route lookup as string matching, in: Proceedings of IEEE Local Computer Networks, 2000, pp. 589–595.

[4]. J. van Luntereb, Searching very large routing tables in wide embedded memory, in: Proceedings of IEEE GLOBECOM, vol. 3, 2001, pp.1615–1619.

[5]. P. Gupta, S. Lin, N. McKeown, Routing lookups in hardware at memory access speed, in: Proceedings of IEEE INFOCOM'98, 1998, pp. 1240–1247.

[6]. K. Zheng, Z. Liu, B. Liu, A scalable IPv6 route lookup scheme via dynamic variable-stride bitmap compression and path compression, J. Computer Communication (29) (2006) 3037-3050.

[7]. M. Degermark, A. Brodnik, S. Carlsson, S. Pink, Small forwarding tables for fast routing lookups, in: Proceedings of ACM/SIGCOMM' 97, 1997, pp. 3–14.

[8]. W. Eatherton, Hardware-based internet protocol prefix lookups, Washington University Electrical Engineering Department (MS Thesis), 1999.

[9]. D.E. Taylor, J.S. Turner, J.W. Lockwood, T.S. Sproull, D.B. Parlour, Scalable IP lookup for Internet routers, IEEE J. Selected Areas Commun. 21 (4) (2003) 522–534.

[10]. S. Kasnavi, P. Berube, V. C. Gaudent, J.N. Amaral, A Multizone Pipelined Chache for IP Routing, LNCS (3462) (2005) 574-585.

[11]. M. Kobayashi, T. Murase, A. Kuriyama, A Longest Prefix Match Search Engine for Multigigabit IP Processing, in: proceedings of IEEE Int'l Conf. Comm. (ICC 00), 2000, pp.1360-1364.

[12]. P. Gupta, Algorithms for Routing Lookups and Packet Classification, doctoral dissertation, Dept. Computer Science, Stanford Univ., 2000.

[13]. V. C. Ravikumar, R. N. Mahapatra, TCAM Architecture for IP lookup Using Prefix Properties, IEEE J. Micro 24 (2) (2004) 60-69.

[14]. H. Liu, Routing Table Compaction in Ternary CAM, *IEEE J. Micro* 22 (1) (2002) 58-64.

[15]. M.J. Akhbarizadeh, M. Nourani, An IP Packet Forwarding Technique Based on Partitioned Lookup Table, in: Proceedings of IEEE Int'l Conf Comm. (ICC 02), 2002, pp.2263-2267.

[16]. R. Panigrahy, S. Sharma, Reducing TCAM Power Consumption and Increasing Throughput, in: Proceedings of 10th Symp. High- Performance Interconnects (HOTI 02), 2002, pp. 107-112.

[17]. F. Zane, G. Narlikar, A. Basu, CoolCAMs: Power-Efficient TCAMs for Forwarding Engines, in: Proceedings of IEEE Infocom 2003, 2003, pp. 42-52.

[18]. R. Lysecky, F. Vahid, On-chip logic minimization, In: Proceedings of the 40th conference on Design automation, 2003, pp 334–337.

[19]. A. Mahini, R. Berangi, H. Mohtashami, M.S. Esfahani, H. Mahini, Storage Efficient and Power Optimized TCAM Based Architecture for Forwarding IP Packet, in: proceeding IEEE International Symposium on Performance Evaluation of Computer and Telecommunication Systems, 2007, pp. 129-136.

[20]. A. Mahini, R. Berangi, H. Mohtashami, S.F. Khatami Firouzabadi, H. Mahini, Low Power TCAM Forwarding Engine for IP Packets, in: proceedings of Int'l Military Communication Conference, Orlando USA, 2007, pp. 1-7.

[21]. A. Mahini, R. Berangi, H. Mahini, H. Mohtashami, Scalable, Low Power and Storage Efficient Parallel lookup Engine Architecture for IP Packets, in: proceedings of 15th Int'l Conference of IEEE Mixed Design of Integrated Circuits and Systems (MIXDES 2008), 2008, pp 611-618.

[22]. C. Labovitz, G. R. Malan, F. Jahanian, Internet routing instability, IEEE/ACM J. Transaction on Networking 6 (5) (1998), 515-528.

[23]. V.C. Ravikumar, R. Mahapatra, and L.N. Bhuyan, "EaseCAM: An Energy and Storage Efficient TCAM-Based Router Architecture for IP lookup", IEEE J. TRANSACTIONS ON COMPUTERS 54 (5) (2005), 521-534.

[24]. Q. Li, T. Jinemi, K. Shima, IPv6 Core Protocol Implementation, Morgan Kafmann, 2007.

[25]. "BGP Reports", URL: http://bgp.potaroo.net/, visited: 8/23/2010.

Authors

Hamidreza Mahini, received the BS degree in software engineering in 2006 from Iran University of Science and Technology (IUST), Iran and he received the MS degree in IT Engineering in 2009 at the same university. Now he is an academic staff at Islamic Azad University Gorgan branch and director of ICT office at the same university .His current researches focus on high performance IP route lookup, network processor architecture and wireless intrusion detection, prevention and response techniques.

Reza Berangi received the PhD in Mobile Telecommunications from Victoria University, Australia. Now he is an Assistant Professor at Iran University of Science and Technology (IUST) and head of the wireless network lab in the school of computer engineering in the same university. His current researches focus on high performance IP route lookup, network processor architecture and mobility management for the next generation networks (NGN).