TCAM-Based Forwarding Engine with Minimum Independent Prefix Set (MIPS) for Fast Updating

Gesan Wang and Nian-Feng Tzeng
Center for Advanced Computer Studies, University of Louisiana at Lafayette
Lafayette, LA 70504-4330, U.S.A.
{gxw2096, tzeng}@cacs.ull.edu

Abstract – Hardware approaches for speedy IP lookups can be realized by making use of TCAMs (Ternary Content Addressable Memories), whose lookups utilize IP addresses as search keys with each search requiring only a single memory access. However, most existing TCAM-based forwarding engines involve shifting TCAM entries when the forwarding table is updated, typically incurring a lengthy update duration. In this paper, a TCAM-based longest prefix forwarding engine with fast updating is proposed. The key idea behind the design is to maintain the forwarding table in a TCAM according to the Minimum Independent Prefix Set (MIPS), totally avoiding the need to shift TCAM entries during updating. Experimental results show that our design is capable of supporting fast TCAM updates, lowering the adverse impact of table updates on IP lookup performance. In addition, our MIPS approach exhibits considerable forwarding table compression.

I. INTRODUCTION

A critical function of the Internet routers is to forward packets by performing route lookups on a forwarding engine (FE) — When an IP packet arrives at a router linecard, the FE locates an entry in a routing table based on the packet’s destination address and then determines the outgoing linecard (and a proper interface) through which the packet is forwarded towards its ultimate destination.

In the past decade, extensive studies have been carried out on quickening routing table lookups. Among them, solutions using TCAMs (Ternary Content Addressable Memories) to achieve fast routing lookups, has received considerable attention lately [1], [2], [3], [4], [5]. TCAM is a fully associative memory device storing wildcards, in addition to 0s and 1s. It is capable of comparing a desired pattern against the entire list of pre-stored entries simultaneously, enabling a search operation over its all entries in a single clock cycle. When compared with its software counterparts like binary or trie-based searches over entries kept in regular SRAM/DRAM [6], [7], TCAM-based approaches often deliver a tenfold or better reduction in search time [8]. As a result, a TCAM-based solution is very attractive for fast routing lookups.

Nonetheless, TCAMs traditionally have their own share of drawbacks, notably:

1) They exhibit lower density and higher prices compared with conventional memory devices.

2) High power consumption increases overall product operational expenses.

3) Slow update operations retard the search procedure.

During the past few years, the cost of TCAMs has been dramatically reduced and density has been vastly increased due to the advances in processing technology and multilevel metallization. Nowadays, very dense, high-speed multi-megabit TCAMs can be fabricated economically. For instance, the IDT’s 32K×75 (2 Mbit) network search engine is priced at only $30 [9].

The second drawback refers to the incremental overall power supply and cooling expense consumed by TCAMs [15]. Initially, high-density TCAM devices consume as much as 12-15 Watts each when all the entries are enabled for search. Fortunately, block select technique and advanced power management remarkably trim the power consumption. For example, SiberCore Technologies claims that its 18 Mbits capacity TCAM product, SCT1842, consumes less than 6 Watts power [10].

The last drawback arises from the fact that instabilities of the Internet require a forwarding table to be updated frequently to reflect its route changes. Especially, a backbone router may experience 100 updates to its routing table per second on an average and the number could reach as high as 1,000 updates per second [11]. When a TCAM is used to accommodate the forwarding table, a single update, either adding or deleting a prefix, involves multiple TCAM entry moves, if the order constraint of the TCAM entries has to be maintained (typically for most known TCAM-based lookup solutions). When a TCAM is updating, search operations over the updating range must freeze until the update operations complete. Consequently, high frequent updating seriously limits a router’s lookup performance. Two fast TCAM update algorithms for LPM have been proposed [12] to reduce the number of entry moves per update by establishing the prefix-length order constraint as well as chain-ancestor ordering constraint and by keeping the empty TCAM entries in the center of a TCAM. One of these algorithms is optimal in terms of the worst-case situation (at most 16 entry moves for IPv4) per update. Although the goal of their design is to minimize the TCAM locking time for updates, the entire TCAM still needs to be blocked from lookups until an update operation has finished. As illustrated in [13], attempting to lock a forwarding table for its updates can significantly lower the performance of search processing. For example, locking

This work was supported in part by the National Science Foundation under Grant CCR-0105529.
an LPM table for 16 prefix moves actually leads to dropping or buffering some 18 packets at the OC-192 line rate.

A Consistent Policy Table Update Algorithm (CoPTUA) has been proposed recently [13], aiming to avoid locking the TCAM during updating while ensuring the consistency of the table by eliminating direct TCAM entry overwriting. However, the algorithm still requires TCAM entry shifts and seems difficult to be implemented.

In this paper, we propose a software accelerated, TCAM-based forwarding engine architecture that performs real-time routing table search with fast dynamic updating. The proposed forwarding engine (1) can locate the longest matched prefix, regardless of its location in the TCAM, and (2) produces only one single matched prefix (rather than multiple ones for typical TCAM-based approaches). This is made possible by converting a CIDR forwarding table into the Minimum Independent Prefix Set (MIPS), according to which TCAM entries are stored. In addition, a special designed trie structure is maintained (and kept in DRAM) to offload the burden of table updating and TCAM memory management off TCAM.

The proposed design is valuable for several reasons:
1) Eliminating the need to sort TCAM entries and to arrange spaces for insertions and deletions.
2) Totally avoiding shifts of TCAM entries upon routing table updating.
3) Achieving fast and consistent incremental updates without locking TCAM.
4) Sustained and fast table search – exactly one TCAM memory access per lookup.
5) Considerable TCAM forwarding table compression.

The rest of the paper is organized as follows. Section II describes the architecture of our proposed search engine. Section III gives some preliminaries and notations. Section IV and V explain how our design works in detail. Experimental results and performance evaluation are presented in Section VI. Finally, section VII concludes this paper.

II. FORWARDING ENGINE ARCHITECTURE

The system architecture of our proposed MIPS Forwarding Engine (MIPS-FE) is depicted in Fig.1. Besides TCAM and its associated SRAM used respectively to accommodate the forwarding prefixes and their corresponding next-hops, there are two other major components: data plane and control plane, in support of forwarding table lookups and updating.

A. Data plane
The data plane takes the IP header of a received packet and passes it to TCAM, where the MIPS forwarding table is searched, with the address of the matched entry delivered to SRAM for retrieving the proper next-hop field. Finally, SRAM returns the next-hop result back to the data plane.

B. Control plane
The control plane translates a received update message into some update operations via an auxiliary 1-bit trie structure maintained therein and passes them to TCAM for incremental updates.

C. TCAM
Although working with conventional 2-port TCAM, our design is described here under TCAM of 3 ports (with two for lookups and one for updating/maintenance). A dedicated TCAM updating port prevents table maintenance from interrupting lookup operations. Besides, 3-port configuration allows the control plane to be separated from the data plane, operating independently without any performance penalty when carrying out MIPS table maintenance such as incremental updates and data compression. While the use of 3-port TCAM also alleviates performance degradation caused by table maintenance for earlier known TCAM-based designs, our design stands to benefit more due to its fewer TCAM accesses per update, as will be demonstrated in Section IV.

D. SRAM
SRAM is used to accommodate next-hop information corresponding to each forwarding prefix in TCAM. It is indexed by the location of the matched prefix in TCAM (as the address of SRAM entry) and returns next-hop information kept in the entry back to the data plane as the search result.

In the remaining of this section, we summarize some hardware features that enable our MIPS forwarding engine to achieve its functionality. Most of these features are commonly supported by TCAM-based search engine products available in the market.

1) In addition to an inbound data port from the data plane and an outbound address port to next-hop memory, TCAM is equipped with a third port operating with the control plane. That separate port is dedicated to TCAM maintenance, permitting table updates to proceed without interrupting or suspending the search path.

2) Each TCAM entry can be tagged with Access Control Status (ACS) that can be any one of “invalid”, “valid”, “hit”, or “updating”. All empty and outdated entries are set to the “invalid” status. Invalid entries are not involved in the search operations (for saving power) but are allowed to be overwritten immediately. Accordingly, the deletion operation is nothing but setting a specified entry to “invalid”. All “valid” entries, if not disabled, are compared in parallel during search operation and allowed to be updated. When a prefix in a “valid” entry is matched, the entry is set to “hit”, disallowing it to be deleted or updated until the address of its corresponding next-hop memory location is latched and then its status is reset to “valid”. This kind of hits is called “exact”
hits. When an entry is under updating, it is in the “updating” status. When an “updating” entry is hit, its corresponding next-hop field is not returned until the update completes. This kind of hits is referred to as “suspended” hits.

3) During a search operation, the search key (an IP address) is placed in a special register, named the Comparand Register (CR). When a “suspended” hit occurs, the key retains in CR until the updating operation finishes. When a new search starts, its key is loaded to CR.

4) If a hit happens, no matter an exact or a suspended hit, a Match Flag (MF) is asserted to announce that the data in CR is found in memory. The Match Flag is reset until the search operation completes, i.e., the address of its corresponding next-hop memory location is latched.

5) A Next Free Entry Register (NFER) is used to keep the address of a TCAM location that is available for accommodating a new prefix. Specifying NFER is simple because the absence of the order constraint in TCAM eliminates the need to rearrange TCAM entries after each deletion operation.

6) In case a prefix is already in a TCAM entry during updating, its corresponding next-hop field in SRAM is modified accordingly, after searching the prefix in TCAM.

It should be noted that ACS, CR, MF, and the NFER are located in the TCAM control logics, as shown in Figure 1.

III. PRELIMINARIES AND NOTATIONS

The IP address space could be considered as a set of consecutive IP addresses in a range from 0 to $2^n-1$, where $W$ is the length of IP addresses (with $W = 32$ for IPv4 and $W = 128$ for IPv6). An IP prefix $p$ can be represented as $\{p_i \mid p_i \in \{0, 1\} \text{ for } 0 \leq i \leq |p|\}$, where $|p|$ denotes the length of prefix $p$ that is equal to the number of non-wildcard bits in $p$. Each prefix $p$ has a corresponding IP address range $\alpha = [a, b]$ that contains a set of consecutive IP addresses $[a, a+1, a+2, \ldots, b]$ ($a \leq b$), where $a$ and $b$ are the minimum and the maximum IP addresses covered respectively by prefix $p$.

Given two prefixes $p$, $q$ and their corresponding ranges $\alpha = [a, b]$ and $\beta = [c, d]$, we provide the following definitions and preliminaries to facilitate further discussion.

Definition 1: If range $\beta$ is totally covered by range $\alpha$, i.e., ($\alpha \cap \beta = \beta$), prefix $p$ is called an ancestor prefix relative to prefix $q$. In particular, if $|p| = |q|-1$, $p$ is called the parent prefix of $q$.

Obviously, if $p$ is an ancestor prefix of $q$, then $|p| < |q|$ and $p_i = q_i$ when $0 \leq i \leq |p|-1$; if $p$ is a parent prefix of $q$, $p$ can be represented as $\{p_i \mid p_i = q_i \text{ for } 0 \leq i \leq |q|-2; p_i = “don’t care” \text{ for } |q|-1 \leq i \leq W-1\}$.

Definition 2: Ranges $\alpha$ and $\beta$ are conjunct if and only if $c - b = 1$; their corresponding prefixes $p$ and $q$ are called conjunct prefixes.

Definition 3: Prefixes $p$ and $q$ are coupled if and only if $|p| = |q|$ and $p_i = q_i$ for $0 \leq i \leq |p|-2$.

Definition 4: In a set of prefixes, if a prefix does not have an ancestor prefix existing in the set, the prefix is called an independent prefix. If a set of prefixes contains only independent prefixes (namely, no prefix can be an ancestor prefix of any other), the set of prefixes is called an Independent Prefix Set (IPS).

Definition 5: In an IPS, if any pair of coupled prefixes has distinct next-hop values, the IPS is called the Minimum Independent Prefix Set (MIPS).

Lemma 1: In a forwarding table, two coupled prefixes that have an identical next-hop value can be replaced by their parent prefix with the same next-hop address without wrecking the correct forwarding functionality.

Proof. Suppose prefixes $p$ and $q$ have the same next-hop value $\eta$; prefix $p$ covers range $\alpha = [a, b]$ and prefix $q$ cover range $\beta = [c, d]$. Any IP address between $a$ and $b$ matching $\alpha$ as the longest-matching prefix will be forwarding through next-hop $\eta$ and any IP address between $c$ and $d$ matching $\beta$ as the longest-matching prefix will also be forwarding through next-hop $\eta$. $p$ and $q$ are coupled prefixes and have a common parent prefix $r$. According to Definition 1, $r$ must cover both range $\alpha$ and $\beta$. After $p$ and $q$ are removed from the table with $r$ and its corresponding next-hop $\eta$ inserted, $r$ becomes the longest-matching prefix for all IP address between $a$ and $d$. Hence, any address previously covered by $p$ and $q$ will still be forwarded through next-hop $\eta$.

Lemma 2: In a forwarding table managed according to MIPS, an IP address can match one and only one prefix.

Proof. This is straightforward, because there exists no prefix which is a prefix of another.

With the above preliminaries, it is clear that transferring an IP prefix table following MIPS can benefit a TCAM-based routing lookup scheme, as a result of the facts below:

1) Keeping prefixes in a strict order in TCAM is no longer compulsory.
2) Shifting forwarding prefixes in TCAM while updating is totally avoided.
3) Empty space management is remarkably simplified.
4) The adverse impact of updating on TCAM lookup performance is significantly reduced.

The next sections give detailed explanation of these facts.

IV. MIPS FORWARDING TABLE PRE-CONSTRUCTION

To support the conversion from an original forwarding table to a MIPS-based equivalence, an auxiliary 1-bit trie is built and maintained in the control plane. Figure 2 shows the 1-bit trie structure adopted to support our approach. The 1-bit trie is a tree-like structure on which each node (called TreeNode) consists of two data units, namely, LeftPrefixNode and RightPrefixNode. The TreeNode is used to represent a pair of coupled prefixes and each PrefixNode is associated with a prefix whose value and length correspond to the path from the root of the trie to the current PrefixNode.

When the 1-bit trie is used to represent an IPv4 forwarding table, PrefixNodes at level $l$ (0 $\leq l \leq 31$) for IPv4) of the trie store all information associated with the prefixes of length $l$. 

...
If the rightmost non-wildcard bit of a prefix is bit 0, information associated with the prefix is stored in the `LeftPrefixNode`; otherwise, its information is stored in the `RightPrefixNode`. Note that bits of a prefix are numbered from the leftmost end, as bit 31. Given a prefix, to find its corresponding location on the trie requires a searching process with multiple branching decisions made. At lever i of the trie, branching decision is made by examining the ith bit of the prefix. If its bit value is 0, the search turns to the left subtrie; otherwise, the search turns to the right subtrie.

A forwarding prefix is kept in a TCAM entry as a `<prefix, next-hop>` pair, where `prefix` is a network address and `next-hop` refers to an outbound port number. Given a forwarding table, building its corresponding 1-bit trie is straightforward. Initially, prefixes in the forwarding table are sorted in a descending order in terms of prefix length to avoid the costly backtracking. A 1-bit trie is constructed by inserting the sorted prefixes into the trie one by one, starting with an empty trie. Eventually, each prefix in the original forwarding table corresponds to a `PrefixNode` on the 1-bit trie, which is called an `ORIGINAL` node. In Figure 2, all `PrefixNodes` marked with bold “P#” represent the forwarding prefixes in the original forwarding table.

Transferring a given forwarding table following the `Minimum Independent Prefix Set` involves two phases:

1) **Expansion** – expand a conventional forwarding table to an `Independent Prefix Set`.

2) **Compression** – compress the expanded table to the `Minimum Independent Prefix Set`.

Expansion can be done easily over the auxiliary 1-bit trie. When a prefix is inserted into a 1-bit trie as an `ORIGINAL` node, the insertion operation continues going down until it “meets” an existing `ORIGINAL` node or a leaf node (as shown by the dashed lines in Figure 2(a)). During this process, all empty `PrefixNodes` inherit the same next-hop value from their common `ORIGINAL` ancestor and are made as `INHERITED` nodes. Finally, the expansion operations yield a set of independent prefixes each of which has an associated leaf `PrefixNode` (either an `ORIGINAL` or an `INHERITED` node) on the 1-bit trie.

An example of expansion is depicted in Figure 2(a), where `PrefixNodes` 101*, 111*, and 1101* inherit the same next-hop P2 from their common `ORIGINAL` ancestor, 1*. They are marked as `INHERITED` nodes.

Decomposing an original prefix table into an independent prefix set can partially satisfy our goal, i.e., eliminating the requirement of keeping prefixes in a strict order in TCAM. However, this decomposition tends to cause a large increase in the number of prefixes in TCAM. To cope with this shortcoming, a remedy for eliminating redundant routing prefixes, known as compression, is adopted as follows. Compression is achieved based on an observation that in an independent prefix set produced in the expansion phase, some `conjunct` prefixes composing a subset of independent set, say \{p\}, covered by a common `ancestor` prefix q have the same next-hop as that of q. Intuitively, all prefixes in \{p\} are redundant in the final forwarding table, and therefore can be replaced by prefix q. Figure 2(b), for instance, prefixes 0100*, 0101*, 0110*, and 0111* on the subtrie rooted at node 01* are all redundant. They can be replaced by prefix 01* and node 11* in the final MIPS forwarding table.

Our compression algorithm is based on **Lemma 1** stated earlier, i.e., two `coupled` prefixes with an identical next-hop can be combined and replaced by their `parent` prefix associated with the same next-hop. This process starts when an expansion reaches a leaf node; it is invoked recursively upward until a `TreeNode` with two different next-hop values is encountered (as shown by the dashed lines in Figure 2(b)). Our experimental results show that the compression algorithm is very efficient in terms of the number of memory accesses and memory utilization (see Section V).

All prefixes corresponding to leaf nodes on a trie are then entered to TCAM. Because no prefix in TCAM is an ancestor of any other prefixes (as indicated by **Lemma 2**), TCAM updating is dramatically simplified, resulting from the facts that: (1) maintaining prefixes in a strict order in TCAM during updating is not compulsory (see Figure 2(c)); (2) an empty entry can be located at any location in TCAM to accommodate a new prefix. In addition, a MIPS-based forwarding table benefits the IP lookup process, since (1) longest prefix match is not necessary as each search in TCAM now and always yields exactly one match and thus no priority encoder is required, and (2) TCAM updating has a reduced impact on search performance for it no longer locks TCAM. It should be noted that when a TACM search is done without a priority encoder, its search latency can be reduced by up to 50% [14].

![Figure 2. Example of MIPS-based forwarding table pre-construction.](image-url)
V. INCREMENTAL UPDATES

Our incremental update algorithm is based on the "expansion then compression" process described above (for initial table construction). However, the update algorithm is more complicated, when compared to initial table construction due to the following reasons: (1) an update comes with arbitrary prefix length, and (2) incorrect prefix matching during the update process has to be avoided.

Updating MIPS-based forwarding table consists of two phases: trie updating and TCAM updating. In order to ensure MIPS in TCAM, an update is processed on the 1-bit trie first to find its effect on TCAM. The effect can range from no TCAM operation to multiple TCAM operations. In other words, one update to the original forwarding table may result in zero or multiple TCAM entry deletions and/or insertions to TCAM. Although the update process on the 1-bit trie is relatively time-consuming, it is taken care by the control plane and does not disturb search operations on TCAM. Due to the space limitation, the general updating process on the trie is not detailed here. Instead, some typical update scenarios will be explained next, as demonstrated in Figure 3.

Figure 3(a) gives an example of inserting a new prefix <101*, P5> to the constructed MIPS forwarding table shown in Figure 2(c). This insertion causes a deletion of an existing prefix <100*, P5> from TCAM and an insertion of a new prefix <10*, P5> to TCAM. PrefixNodes 100* and 101* are pruned from the trie since they have an identical next-hop, P5.

Figure 3(b) shows the insertion of new prefix <010*, P6> to the MIPS-based table depicted in Figure 3(a). The next-hop P6 is pushed down to node 0101* and next-hop P3 is pushed down to node 011*. Hence, this insertion causes existing prefix <01*, P3> to be deleted and two new prefixes, <011*, P3> and <0101*, P6>, inserted to TCAM. Part of the subtrie rooted at node 01* is reactivated.

Figure 3(c) demonstrates the deletion of prefix <0*, P1> (corresponding to the ORIGINAL node 0*) from the original forwarding table. This deletion results in an existing prefix, <001*, P1>, removed from TCAM as the next-hop of node 001* is inherited from node 0* (see Figure 3(b)).

Figure 3(d) depicts the deletion of prefix <01*, P3> from the original forwarding table, leading to a deletion of prefix <011*, P3> and an addition of prefix <0111*, P3> to TCAM.

Figures 3(e) and 3(f) show that updating the next-hops of prefixes in a forwarding table may cause both prefix additions and deletions in TCAM. Some cases may not involve any insertion or deletion in the table. For instance, adding prefix <0101*, P3> to, or deleting prefix <0100*, P3> from Figure 2(c) will not affect any existing TCAM entries.

Due to the salient features of our MIPS-based forward engine described above, updating a single prefix in TCAM is pretty simple because all the prefixes are accommodated without any order constraint. More specifically, a new prefix can be inserted to any available location in TCAM pointed by the Next Free Entry (NFG) register. An existing prefix can be deleted by simply setting its corresponding TCAM entry to "invalid" status without bothering all other entries. Modifying the next-hop field of an existing prefix involves a TCAM search plus a SRAM write. On the other hand, an update to the original forwarding table may result in multiple TCAM operations, be an insertion, a deletion, or a change to the next-hop field.

Figure 3. Some typical examples of incremental updating.

It is undesirable to lock the forwarding table upon updating in the presence of multigigabit line rates, due to potentially significant impact on lookup performance. However, if the table is not locked during updating, two possible types of incorrect prefix matching could result during the update process [13]: "inconsistent prefix matching" and "erroneous prefix matching". The former means the matched prefix is not with the longest prefix, whereas the latter signifies the matched prefix is partially updated.

Fortunately, inconsistent prefix matching can be avoided in our MIPS-FE by carrying out TCAM updating in two phases: deleting all prefixes which do not appear in the final MIPS-based table first, and then adding new prefixes caused by the expansion and compression operations. Clearly, after each individual prefix deletion and addition during the two phases, all the prefixes in the partially updated table still remain independent, calling for no rearrangement. Any matched prefix of an IP address is always the correct one. Therefore, IP lookup process can proceed and the consistency of the MIPS table is guaranteed.
To tackle erroneous prefix matching, constraints are put to the next-hop update process and the search process. Specifically, the next-hop of a “hit” entry is not allowed to be updated until the search process completes. When an “updating” entry is hit, its corresponding next-hop field is not returned to the data plane until the entire update to the entry is done. The TCAM search process during the “suspended” hit is put off to ensure the return of a correct next-hop result. These two constrains guarantee error-free prefix matching.

VI. EXPERIMENTAL PERFORMANCE EVALUATION

Sixteen routing tables had been downloaded from different sources [15], [16], and [17] and employed in our extensive simulation. They include ten large size tables (> 100,000 entries), four medium size tables (> 20,000 entries), and two small size tables (< 10,000 entries) for demonstrating the generality of our design.

Table 1 lists the sizes of those original sixteen tables and of their corresponding MIPS-based tables in terms of the number of prefixes. As can be seen, all the MIPS-based tables are smaller than their original counterparts. The compression ratio is seen to range from roughly 20% to more than 68 %, and it is irrelevant to the sizes of original routing tables in general. Since the performance of a MIPS-FE is proportional to the number of memory accesses, Table 1 includes that parameter, which consists of two types of memory access components: accesses to the trie structure and accesses to TCAM. The first access component results are listed in the 5th column, indicating the average number of memory accesses on the 1-bit trie per MIPS-based forwarding table update, ranging from 21.87 to 25.36 memory accesses. This is expected since there is no prefix in these routing tables with length shorter than 8 (bits) and the majority of prefixes (> 70%) have length between 20 and 24. Hence, if an eight-bit node is used as the trie root, the average number of memory accesses is expected to lie between 14 and 18. If a single memory access time on the trie is assumed to be 10 ns (when the trie is maintained in off-chip SRAM), the program execution time is 500 ns (for executing some 500 instructions per update, where instructions are brought to the I-cache with a mean access time of 1 ns per instruction therein), and the average number of trie accesses is 25 per update, then, one update to the trie amounts to roughly 10×25+500 = 750 ns. In other words, the MIPS-based trie is able to handle about 1.3 million updates per second, if devoted fully for updating, or equivalently, it allows more than 99% of the network processor’s computing power for other tasks while supporting 10,000 table updates per second (at least 10 times more than what are likely to happen to a current backbone router).

Column six of Table 1 displays the mean number of TCAM accesses per MIPS table update. It ranges from 1.05 to 2.03 memory accesses per update, and it seems independent of the original table size. This number is in contrast to earlier results presented in [13], where the average number of TCAM entry movements for the PLO_OPT (or the CAO_OPT) algorithm is about 4 (or 1.02 to 1.06) per update. When comparing our MIPS approach with the earlier algorithms, however, one has to note that a TCAM entry movement actually involves one TCAM search operation plus one TCAM write operation. In other words, the PLO_OPT (or CAO_OPT) algorithm requires roughly 8 (or 2.04 to 2.06) TCAM operations per update, rendering our MIPS-FE clearly superior. It should be noted that PLO_OPT and CAO_OPT algorithms also make use of auxiliary 1-bit tries similar to ours, but the time required for trie accesses was not dealt with therein [12].

<table>
<thead>
<tr>
<th>AS Number</th>
<th># of initial prefixes</th>
<th># of prefixes in MIPS (TCAM)</th>
<th>Compress. rate (%)</th>
<th># of memory accesses in trie</th>
<th># of TCAM accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS1221</td>
<td>181295</td>
<td>80297</td>
<td>55.71</td>
<td>25.36</td>
<td>1.42</td>
</tr>
<tr>
<td>AS12654</td>
<td>170459</td>
<td>105560</td>
<td>38.13</td>
<td>23.40</td>
<td>1.63</td>
</tr>
<tr>
<td>AS6447</td>
<td>166839</td>
<td>120998</td>
<td>27.48</td>
<td>22.62</td>
<td>1.97</td>
</tr>
<tr>
<td>AS7660</td>
<td>158298</td>
<td>492299</td>
<td>68.68</td>
<td>23.87</td>
<td>1.05</td>
</tr>
<tr>
<td>AS6539</td>
<td>157684</td>
<td>121907</td>
<td>22.69</td>
<td>23.82</td>
<td>1.55</td>
</tr>
<tr>
<td>AS3333</td>
<td>154286</td>
<td>79484</td>
<td>48.48</td>
<td>23.92</td>
<td>1.40</td>
</tr>
<tr>
<td>AS286</td>
<td>148466</td>
<td>113945</td>
<td>23.23</td>
<td>23.51</td>
<td>2.03</td>
</tr>
<tr>
<td>AS4637</td>
<td>147702</td>
<td>53037</td>
<td>64.09</td>
<td>23.91</td>
<td>1.28</td>
</tr>
<tr>
<td>AS15290</td>
<td>144029</td>
<td>111538</td>
<td>22.56</td>
<td>23.88</td>
<td>1.54</td>
</tr>
<tr>
<td>AS3277</td>
<td>134993</td>
<td>94833</td>
<td>29.75</td>
<td>24.09</td>
<td>1.59</td>
</tr>
<tr>
<td>AS5459</td>
<td>78133</td>
<td>62432</td>
<td>20.09</td>
<td>23.63</td>
<td>1.51</td>
</tr>
<tr>
<td>AS3303</td>
<td>68739</td>
<td>52375</td>
<td>23.80</td>
<td>21.87</td>
<td>1.45</td>
</tr>
<tr>
<td>AS1741</td>
<td>41435</td>
<td>19722</td>
<td>52.40</td>
<td>23.53</td>
<td>1.19</td>
</tr>
<tr>
<td>AS267</td>
<td>21112</td>
<td>13876</td>
<td>34.27</td>
<td>22.89</td>
<td>1.52</td>
</tr>
<tr>
<td>AS10764</td>
<td>9495</td>
<td>6678</td>
<td>29.67</td>
<td>23.31</td>
<td>1.20</td>
</tr>
<tr>
<td>AS11537</td>
<td>9430</td>
<td>6920</td>
<td>26.62</td>
<td>23.16</td>
<td>1.19</td>
</tr>
</tbody>
</table>

When the MIPS-FE starts its operation, its routing/forwarding table size enjoys considerable reduction according to our simulation results, as revealed in Table 1. Since the forwarding engine keeps receiving updates to its routing table during the course of its operation, we examined how the routing table size got changed under the MIPS approach when incremental updates progressed. Various routing tables and their associated update files were downloaded for examination. Each update file includes mainly prefix additions (namely, new prefixes inserted or next hop changes to existing prefixes), plus prefix withdrawals. Our examined results indicate that the MIPS approach leads to substantially fewer entries changed and achieve rather consistent compression over the entire incremental updating period, for any initial routing table and its associated update file obtained for examination. As an instance, Figure 4 illustrates snapshots of over 50,000 incremental updates (contained in the update file) to a routing table (rrc00) downloaded from RIPE Network Coordination Center (RIPE-NCC) [17]. (The update file was also obtained from RIPE-NCC.) Initially, rrc00 contains 170,459 prefix entries. At the beginning, its corresponding table established following MIPS has 38.13% fewer entries than the original
one. The update file consists of 43,327 prefix additions (either new prefixes inserted or next-hop updates to existing prefixes) and 6,673 prefix withdrawals. It can be observed from Figure 4 that the MIPS table maintains consistent table size reduction throughout its updates. While the original table size rises consistently from 170,459 to more than 178,300 entries, the corresponding MIPS table experiences a much smaller increase in it size (from about 105,460 to 106,200 entries or so), keeping the compression rate always around 39% or better.

By analyzing a long term updating trace over rrc00, we also found that a prefix withdrawal normally caused more TCAM accesses than a prefix addition. For example, the average number of TCAM accesses per withdrawal is 1.26, as opposed to 0.69 for each addition. This phenomenon can be explained by the fact that an added prefix may “see” one (or multiple) existing prefix(es) with an identical next-hop value as its ancestor(s) located on the 1-bit trie, therefore requiring no access to TCAM following our MIPS approach. It is especially advantageous to have lighter TCAM accesses for prefix additions, as they often account for the vast majority of prefix updates, making the proposed approach exhibit far less lookup performance degradation during incremental updates.

VII. CONCLUSION

TCAM-based IP address lookup solutions are very attractive for high-performance router construction. In spite of the advantage of fast lookups, however, such a hardware solution usually suffers from slow updating, potentially hampering lookup performance substantially. This paper has introduced a new TCAM-based search engine architecture realized by following the Minimum Independent Prefix Set (MIPS) aiming to effectively simplify updating in the TCAM-based forwarding table. As a result, frequent table updates no longer have a marked adverse impact on lookup performance. Additionally, our MIPS-based approach enjoys respectable table compression that lowers the table size by more than 20%.

REFERENCES