CLUE: achieving fast update over compressed table for parallel lookup with reduced dynamic redundancy

Tong Yang, Ruian Duan, Jianyuan Lu, Shenjiang Zhang, Huichen Dai and Bin Liu
Dept. of Computer Science and Technology, Tsinghua University, Beijing China

***Abstract*—The size of routing tables in backbone routers continues to keep a rapid growth, and an effective solution is routing table compression. Meanwhile, there is an increasingly urgent demand for fast routing update due to the increase of mobile communications and new emerged Internet functionalities. Furthermore, the link transmission speed of backbone routers has increased up to 100Gbps commercially and towards 400Gbps Ethernet for Lab experiments, resulting in a raring need of ultra-fast routing lookup. Therefore, to achieve high performance, backbone routers must gracefully handle the three problems: routing table Compression, fast routing Lookup, and fast incremental UpdatE (CLUE). Previous works often only concentrate on one of the three dimensions, failing to satisfy the future high speed requirement.**

To address these issues, we propose a complete set of solutions—CLUE, by improving previous works of our group and adding a novel incremental update mechanism. CLUE consists of three parts: a routing table compression algorithm, an improved parallel lookup mechanism, and a new fast incremental update mechanism. The routing table compression algorithm is based on our previous work: ONRTC algorithm [24], which is a preparation for fast TCAM parallel lookup and fast update of TCAM and redundancy. The second part is the improvement of the logical Caching scheme for dynamic Load-balancing Parallel Lookup (CLPL) mechanism [1]. And the last one is the conjunction of the trie, TCAM and dynamic redundancy update algorithms. Mathematical proof shows that speedup factor is proportional to hit ratio of redundancy in the worst case, which is confirmed by our extensive experiments. Large-scale experimental results show that, compared with CLPL, CLUE only needs about 71% TCAM entries, 4.29% update time and 3/4 dynamic redundancy for keeping the same high throughput. In addition, CLUE outperforms CLPL in the following two aspects: the frequent interactions between control plane and data plane caused by redundancy update are avoided, and the priority encoder of TCAM is no longer needed.

I. INTRODUCTION

Internet has maintained a rapid growth for many years, which brings three major problems: i) routing table compression -- due to an annual increase of about 15% of the routing table size, ISPs struggle to suppress the table growth, in order to further postpone the requirement for upgrading the infrastructure; ii) routing lookup -- to handle the current gigabit-per-second traffic, the backbone routers must be able to forward hundreds of millions of packets per second, thus bringing huge pressure to routing lookup; iii) fast update -- facing more and more frequent routing updates, routing table must be incrementally updated as fast as possible. These three problems of Compression, Lookup, and UpdatE are abbreviated to CLUE in this paper, and CLUE also stands for a set of solutions for the three problems.

With regard to i), the typical solutions are [3-6]. Although they can achieve high compression ratio, they can't achieve fast updates, especially when TCAM is used. Therefore, we proposed the ONRTC algorithm in [24], which supports parallel lookup and fast update.

With regard to ii), TCAM-based solutions are usually adopted in backbone routers. Ternary Content Addressable Memories (TCAMs) [7] are fully associative memories that allow a "don't care" state to be stored in each memory cell in addition to 0s and 1s. One TCAM access can finish a routing lookup operation. However, the TCAM-based solutions are of power consumption and expensive cost. In order to achieve power efficiency, F. Zane et al. proposed a bit-selection architecture [7], which hashes a subset of the destination address bits to a TCAM partition, thus making TCAM-based routing tables power efficient. This algorithm is referred to as ID-bit partition algorithm in this paper. It only concentrates on reducing power consumption, but cannot improve the lookup speed.

Today, Ethernet link with 100Gbps has been put into service. Suppose that each packet is at its minimum size of 64 bytes, router should complete a packet lookup every 5.12ns, while common TCAM needs 15~30ns for one lookup. Therefore, parallel lookup with multi-TCAM is destined if TCAM solution is adopted. Toward this target, previous works of our group have proposed two mechanisms: Kai's algorithm [8] and Dong's algorithm [1].

In [8], Kai Zheng et al. proposed a TCAM-based parallel architecture, which employs an intelligent partitioning algorithm, takes advantage of the inherent characteristics of Internet traffic, and increases packet forwarding rate multiple times over traditional TCAMs. Power consumption is also reduced by using ID-bit partition algorithm. In order to achieve load balance, Kai Zheng added 25% redundancy to the system based on the long period statistical results. This algorithm is referred to as Statistical Load-balancing Parallel Lookup (SLPL) in this paper.

However, according to Dong Lin's data mining results [1], the average overall bandwidth utilization was very low but the Internet traffic could be very bursty. Hence, during bursty periods, mapping routings table partitions into TCAM chips based on long-term traffic distribution cannot balance the workload of individual TCAM chip effectively. Therefore, the redundancy should be dynamic, and Dong Lin et al. proposed a power-efficient parallel TCAM-based lookup engine with a distributed logical Caching scheme for dynamic Load-balancing Parallel Lookup (CLPL), which is referred to as CLPL in this paper. By logical caches, the traffic can be allocated to each TCAM relatively evenly.

With regard to iii), current solutions mostly either only focuses on i), or only on ii), but seldom emphasize update, maybe because the update was not so frequent before. However, according to our data mining results, the update problem is becoming more and more serious: the updates messages of the backbone routers have reached 35K per second in the traffic peak, which makes the traditional algorithm not applicable.

The system performance will be optimized, only if the three problems are solved simultaneously. In order to reach this goal, besides the previous works of our group, there are still several points which need to be improved and finished.

1) The routing table should be compressed to save hardware cost.

2) The size of redundancy should be further reduced.

3) The frequent interactions between data plane and control plane should be reduced, or even avoided.

4) The update algorithm should be studied profoundly.

Towards the above four targets, we proposed a complete set of solutions -- CLUE. CLUE consists of a routing table compression algorithm -- ONRTC, an improved parallel lookup mechanism based on CLPL, and a new incremental update mechanism, which involves the trie, TCAM and redundancy update. The design philosophy of CLUE is that we should not view the three problems isolatedly and statically, so as to avoid one-sidedness. In other words, only an organic combination of the three aspects can make the forwarding plane of routers work well, thus CLUE emerges as required.

The first part of CLUE, i.e., the previously proposed ONRTC algorithm [24], compresses the routing table size to 70% of its original size. More importantly, it also benefits the second and the third operation greatly, for prefix overlap is eliminated.

The second part of CLUE is an improvement of CLPL. After compressed by ONRTC, the routing table is not overlapped any more, and thus bringing a lot of advantages to TCAM lookup: 1) the domino effect in the TCAM update process will never happen again; 2) the priority encoder is no longer needed -- this not only reduces hardware cost, but also reduce delay of TCAM lookup time; 3) TCAM partitions can be of the same size without redundancy. 4) The compression has saved a part of TCAM's hardware resource. In addition, we make the following improvements: 1) Dynamic Redundancy (DRed) i doesn't cache TCAM i's prefixes, for TCAM i and DRed i will never be both looked-up simultaneously, and thus 1/4 TCAM space can be saved when using four TCAMs. 2) Because of Longest Prefix Match (LPM), when DRed is missed, the destination IP address should be sent to control panel to compute the prefix which should be stored in DRed by using RRC-ME algorithm [9]. Because overlap is eliminated by ONRTC, RRC-ME algorithm is no longer needed in CLUE. We just need to put the prefix which hits the TCAM into DRed. Therefore, the interactions between control panel and date panel caused by DRed update can be totally avoided.

The third part of the CLUE is a brand new whole update algorithm, including trie update, TCAM update, and DRed update. In order to evaluate the update performance accurately and objectively, TTF (Time to Fresh) is defined here. TTF means the average computing time to update a message, which includes TTF1 (TTF-trie): update time of the trie, TTF2 (TTF-TCAM): update time of TCAM, and TTF3 (TTF-Dred): update time of redundancy. TTF indicates a router's sensitivity to the changes of the network state. The smaller the TTF is, the more sensitive the router will be.

To summarize, the primary contributions of this paper lie in the following aspects:

- We propose an integrated problem – CLUE and a solution set with the same name. CLUE involves routing table compression, parallel lookup, and fast update.
- We present a complete mathematical proof to bound the speedup factor, and verify the mathematical conclusion by experimental results.

The remaining parts of the paper are organized as follows. Section II surveys the related work. An improved mechanism based on CLPL is elaborated in Section III. Section IV illustrates the fast incremental update algorithm of the whole system. And extensive evaluation on CLUE over a large-sized real trace is conducted in Section V. Finally we conclude this paper in Section VI.

II. RELATED WORK

As mentioned above, three major problems are involved: routing table compression, fast lookup, and fast incremental update.

A. Compression Algorithm

With respect to compression algorithm, many researches have been conducted to compress routing table. ORTC [3] was proven to be the theoretical optimal compression algorithm, and it remains the best algorithm in terms of its compression ratio since 1999. However, it has a high computational complexity and poor update performance, which makes it ill-suited to handle the increasing demand for fast incremental updates. In [4], Xin Zhao et al. presented 4-level algorithm at the cost of changing the forwarding behavior of routers. What's worse, the 4-level algorithm could potentially trigger the 'routing table fluctuation' and 'garbage roaming' problems, which are illustrated in detail in [24]. In [5], the binary trie was changed into the trigeminal trie, which corresponds to the three-state properties of TCAM. Therefore, a high ratio might be achieved. However, the update messages probably introduce domino effect. To improve compression ratio, Qing Li et al. adopted suboptimal routing [6]. Nevertheless, it could potentially introduce serious traffic congestion.

All the algorithms mentioned above only focus on routing table compression. If the compressed routing tables are not stored in TCAM, the advantages of compression will not be so evident. When the compressed routing tables are stored in TCAM, there is still a big obstacle of TCAM update -- prefix overlap, due to the introduction of CIDR. Prefix overlap refers to some prefixes are a part of others. This brings many difficulties for routing lookup:

1) Layout in TCAM: Prefixes must be ordered by their length, and then stored in TCAM. Meanwhile, a priority encoder is indispensible.

2) Update Handling: When updates occur, many prefixes will probably move, which is called domino effect in this paper. Although many algorithms manage to resist domino effect, redundancy must be introduced. Redundancy means larger TCAM, more hardware cost, and more power consumption. Nevertheless, domino effect still exists.

3) Power Consumption: one major shortcoming of TCAM is its high power consumption. A very effective solution is TCAM partition. There are mainly two partition algorithms: ID-bit partition [6, 7] and range partition [9]. However, the algorithm to determine the partitions was not presented in [9]. So Lin Dong et al. proposed sub-tree partition [8] to implement range partition. ID-bit partition algorithm cannot split the table evenly. Sub-tree partition can work better, but it will introduce redundancy.

If overlap is eliminated, those problems of TCAM-based scheme can be handled perfectly: 1) prefixes can be stored in TCAM randomly; 2) the priority encoder is no longer needed. 3) domino effect will never happen; 4) TCAM partition can be strictly evenly, without redundancy.

There are several approaches to reduce overlap, such as [10] and [11]. In [10], the routing table is divided into two parts: the overlapping part and the non-overlapping part. This approach can only reduce overlap. As far as we know, only leaf-pushing proposed in [11] algorithm eliminates overlap totally. Unfortunately, leaf-pushing causes routing table expand too much.

Therefore, we have proposed Optimal Non-overlap Routing Table Constructor (ONRTC) algorithm [24]. As shown in Figure 1, any trie can be compressed fast according to these six models, which cover all election situations. Although the models seem to be complicated, the whole algorithm requires only one post-order traversal, which is of high efficiency. The details of the six models are left in [24].

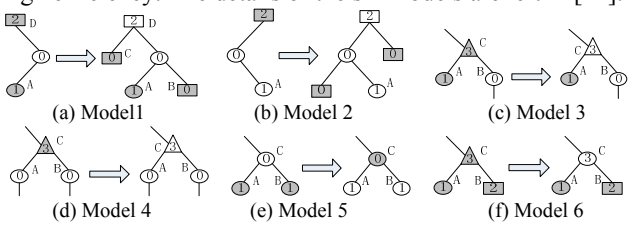


Figure 1. Atomic equivalent models of ONRTC.

B. Routing Lookup Algorithm

With respect to routing lookup, without any load balancing mechanisms, eight or more TCAM chips were employed for parallel lookup operations in [12]. But only a speedup factor of five was achieved if the lookup requests were not evenly distributed. In pursuit of an ultra-high lookup throughput, Zheng Kai et al. proposed a load balancing mechanism [8] based on some 'pre-selected' redundancy. Kai exploits the assumption that the lookup traffic distribution among IP prefixes can be derived from the traffic traces. Then a greedy algorithm with 25% more TCAM entries is proposed to optimize the distribution of these groups among four TCAMs. However, in the worst case, when the traffic is temporarily or permanently biased to a limited number of individual route prefixes, the average throughput may decrease.

Based on the data collected from [13], Lin Dong et al. observed that the average overall bandwidth utilization was very low but the Internet traffic could be very bursty. In order to handle the worst case, CLPL uses logical caches (which actually mean Dynamic Redundancies (DRed)) in place of the static redundancy to achieve dynamic load balance.

C. Incremental Update Mechanism

With respect to incremental update, there are three aspects: 1) trie update; 2) TCAM update; 3) DRed update, if DRed is used.

As for trie update, because SLPL and CLPL adopt no compression algorithm, thus their trie update is fastest; while ONRTC algorithm is adopted in this paper, and the detailed incremental update algorithm is elaborated in [24].

As for TCAM update, one big obstacle is domino effect, and many researchers strive to reduce it.

In [14], the routing table is split into partitions according to the next hop. Each partition holds a collection of all the prefixes which own the same next hop, thus there is no need to keep the prefixes sorted in one partition. Therefore, domino effect can be reduced. However, one more prefixes still be matched, thus the priority is also needed. In addition, it cannot achieve power efficiency.

In [15], all prefixes are split and allocated into different single-match TCAMs based on the ancestor-descendant relationship among them. It presents an algorithm to guarantee that each single-match TCAM generates at most one match for a given destination IP address. However, when a new prefix is inserted, each single-match TCAM may be required to move some of its existing prefixes to another TCAM in order to maintain a disjoint set. What's worse, power efficiency cannot be achieved.

As for DRed update, CLPL adopts LRU policy. But with regard to routing update, SLPL and CLPL didn't mention it. Therefore, in [16], Bin Zhang et al. changed CLPL from the following two aspects:

1) A TCAM-chip is used as a DRed in place of CLPL's four small logical caches. It is evident that this change degrades the whole system's performance. This can be confirmed again by his experimental results in Figure 8 in [16]. The experiments use 12 TCAM chips, and half of the experimental results show that the speedup factor is less than 11, while 11 is the result of CLPL's worst case.

2) Bin Zhang et al. argued that SLPL and CLPL neglected an important factor—the update time. Thus Bin Zhang et al. put the overlapping part of prefixes on the top of the TCAM and at the bottom of the next TCAM, while put the non-overlapping part in the middle of the TCAM. This approach can reduce the domino effect to a certain extent. In contrast, in this paper we eliminate domino effect totally, so as to minimize the negative effect which updates bring.

Therefore, we propose CLUE to handle these problems. As far as we know, this is the first effort on a full-dimensional system optimization for the three major routing table issues. Only focusing on one of them will encounter

short-board effect. Therefore, we combine the solutions organically, and propose a complete set of solutions named CLUE to handle the three major problems simultaneously, so as to optimize the performance of routing forwarding.

III. PARALLEL ROUTING LOOKUP MECHANISM

A. Partition Algorithm

In order to achieve parallel lookup, the prefixes should be split into partitions firstly. Lin Dong's sub-tree partition [1] algorithm outperforms ID-bit partition algorithm. Unfortunately, it will introduce redundancy.

Because prefix overlap is eliminated by ONRTC, partition algorithm becomes very easy, and redundancy is no longer needed. Our new partition algorithm consists of two steps:

Step 1: compute the partition size. Suppose the size of routing table is M and the partition count is n, then the size of each partition is M/n.

Step 2: traverse the trie by inorder, then put every M/n prefixes to each bucket.

It can be concluded that our new partition algorithm is much easier and faster than ID-bit partition algorithm and sub-tree partition algorithm, so as to ensure high memory utilization. Subsequent experimental results also prove this.

B. Parrallel Lookup Mechanism

The detailed implementation architecture of the parallel lookup engine is presented in Figure 2, which is an improved architecture base on CLPL mechanism. The process of our mechanism is shown below (each step is marked by the number in Figure 2):

Step I, when an IP packet arrives, the IP address is sent to the Indexing Logic.

Step II, the Indexing Logic returns a partition number which tells the 'home-TCAM' containing the matching prefix.

Step III, a tag (the sequence number) is attached to the IP address, because this mechanism may cause disorder.

Step IV, the IP address with Tag and partition number is delivered into the Adaptive Load Balance Logic.

Step V, this is the most important step of this mechanism -- Dynamic Redundancy for Load Balancing. To be specific, each TCAM is split into buckets, and one bucket is used as dynamic redundancy (DRed), as shown in Figure 2. The work mechanism of Dynamic Redundancy for Load Balancing is as follows:

a) If the queue of its home TCAM is not full, the incoming IP address will be looked-up in its home TCAM, then the lookup will be finished smoothly; meantime, the other dynamic redundancies will be updated;

b) If the queue of its home TCAM is full, then the incoming IP address will be sent to the idlest queue. One important point is worth mentioning: this kind of IP address will be only looked-up in the related DRed, NOT the home TCAM. Therefore, no IP address will be looked-up in both home TCAM and the corresponding DRed. Base on this sharp observation, DRed i of CLUE doesn't store TCAM i's prefix. By this way, CLUE needs smaller redundancy, but the same hit rate compared to CLPL.

c) If incoming IP address misses DRed, it will be sent back and a) happens again.

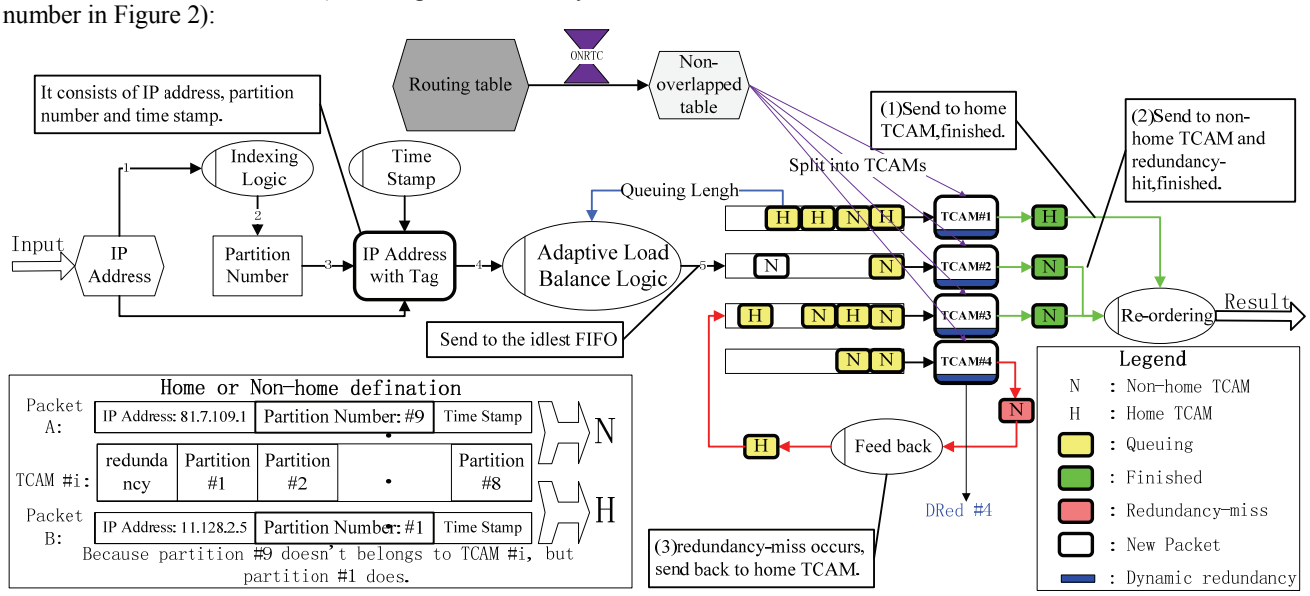


Figure 2. Improved parallel lookup mechanism[1]

C. Novel Dynamic Redundancy Mechanism

As mentioned above, no IP address will be looked-up in both home TCAM and the corresponding dynamic redundancy (DRed). As a result, this is NOT really a cache, and we use the word 'DRed' in place of CLPL's logical cache, but the DRed is updated by cache mechanisms.

Generally speaking, cache mechanisms can be divided into two categories: caching destination addresses (IPs) [17-19] and caching prefixes [20]. Many papers [17-20] have

[1]This parallel mechanism is an improvement of CLPL, which is originally shown in Figure 9 in [1].

demonstrated that caching prefixes is more efficient, and this is also in accord with our experimental results. Therefore, we adopt caching prefixes, as CLPL did.

However, one big obstacle of caching prefix is overlap. Because of LPM, if an inner node is matched, the corresponding prefix cannot be sent to DRed.

For example, as shown in Figure 3, a prefix = 100000 is looked-up in the home TCAM, and the LPM result returns p (p=1*). However, p cannot be sent to DRed, because its child q owns a different next hop. It is obvious that p'=100* should be sent to DRed. This approach is called RRC-ME algorithm [20], which is adopted in CLPL.

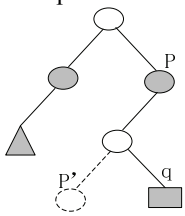


Figure 3. RRC-ME algorithm

For RRC-ME algorithm, one important issue is worth being mentioned here. Although RRC-ME algorithm is simple, its update algorithm is not an easy task. The update algorithm mentioned here means that when the routing table updates, the DRed must updates as well. This process must visit SRAM several times, which is an additional overhead.

In addition, CLPL adopts RRC-ME algorithm, which causes frequent interactions between data plane and control plane, while CLUE doesn't, and the details are as follows.

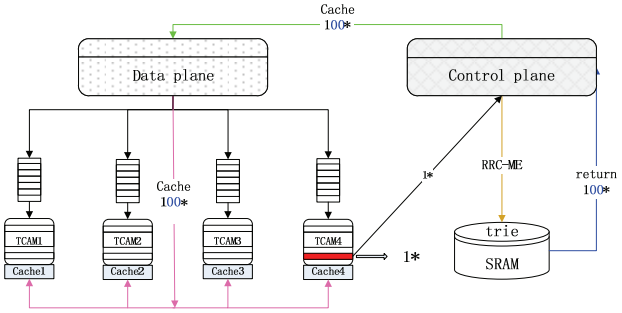


Figure 4. The DRed update process of CLPL's mechanism

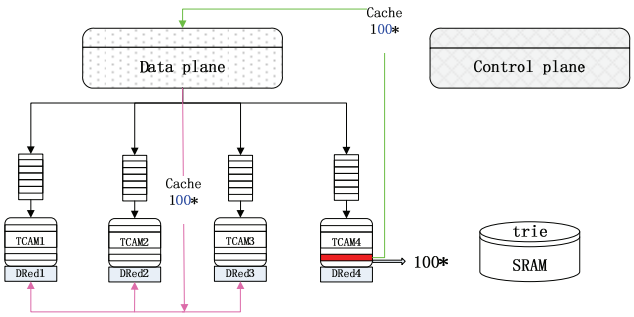


Figure 5. The DRed update process of CLUE's mechanism

The DRed update process of CLPL is shown in Figure 4. If an incoming IP is looked-up in TCAM1, then the LPM prefix is 1*. Prefix 1* must be sent to control panel, which executes RRC-ME algorithm by traversing the trie tree stored in SRAM, and returns 100*. Then 100* is sent to data panel, and is inserted into the four logical caches. It can be noted that the DRed update process must execute RRC-ME algorithm, and frequent interactions happen, disturbing routing lookup a lot.

As aforementioned, DRed i and TCAM i will never be looked-up simultaneously. So CLUE reduces CLPL's DRed size by the rule that DRed i doesn't cache TCAM i's prefixes, and hit rate doesn't decline. With regard to four TCAM chips, the DRed size of CLUE is reduced into 3/4 of that of CLPL.

Because overlap is eliminated, RRC-ME algorithm is no longer needed. Our new DRed process is shown in Figure 5. If an incoming IP address is looked-up in TCAM1, the result of LPM is 100*, then 100* is sent to data panel, and is inserted into the other three DReds. As a result, control panel will not be involved in the process of DRed update process. In this way, the update process of DRed is faster and more efficient.

To sum up, our new DRed Mechanism can achieve smaller DRed size and avoid frequent interactions between data panel and control panel.

D. Lower Bound of the System Performance

With regard to the lower bound of the system performance, Dong Lin et al. has presented a formal mathematical proof, which is flawed. Therefore, a complete formal mathematical proof is given here, and the proof conclusion is in accord with the corresponding experimental results.

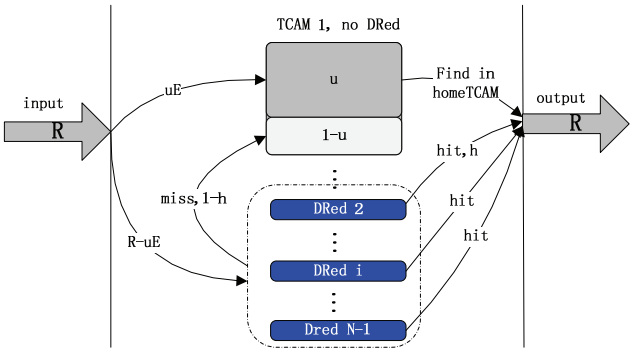


Figure 6. Parallel lookup in the worst case

Our deduction is under the following two premises:

1) The update cost is ignored.

2) The TCAM1 is always working.

The two premises are practically feasible. Regarding premise 1), Dong Lin et al. show that when the cache size is set to 1024 and only one cache-missed element is updated within a 5000 clock cycles, the system can still easily achieve 100% throughput [1]. In addition, each routing update only causes one shift in TCAM, that is, *O(1)* time complexity by CLUE. Therefore, the time cost of routing update is negligible. Premise 2) holds true via a policy that keeps the queue of TCAM1 never empty.

In the worst case of this parallel system, all the traffic is delivered to a single TCAM, which is TCAM1 in Figure 6. It suggests that $TCAM_2$~$TCAM_{N-1}$ is idle, only $Dred_2$~ $Dred_{N-1}$ are still working. The definitions of the symbols are described below:

N means the number of TCAM chips used in this system;

R means the maximum input traffic;

E means the processing ability of each TCAM;

u means the percentage of TCAM1's processing ability used to handle the traffic which goes directly to TCAM1;

1-u means the percentage of processing ability used to handle packets missed in Dred;

h means the hit ratio of the Dred;

t means the speedup factor of this system.

Firstly, it is easy to get each symbol's range.

$$\begin{cases} 0 < u < 1 \\ 0 < h < 1 \\ 2 \le N \end{cases}$$

Secondly, when all TCAMs work at their best, the system can handle the maximum workload R.

$$\left.\begin{array}{l} R = tE \\ uE + (N-1)E = R \end{array}\right\} \Rightarrow t = N + u - 1 \quad (1)$$

$$\Rightarrow t \ge N - 1 \quad (2)$$

Thirdly, TCAM1 preserves 1-u processing ability to handle Dred-missed traffic.

$$\left.\begin{array}{l} R = tE \\ (R - uE) * (1 - h) = (1 - u)E \end{array}\right\} \Rightarrow h = \frac{t-1}{t-u} = \frac{t-1}{N-1}$$

$$h = \frac{t-1}{t-u} = \frac{t-1}{N-1} = \frac{N-2+u}{N-1} \quad (3)$$

$$\Rightarrow h \ge \frac{N-2}{N-1} \quad (4)$$

According to (1) and (3), we get that

$$t = (N-1)h + 1 \quad (5)$$

Then

$$t \ge N - 1$$

As long as

$$h \ge \frac{N-2}{N-1}$$

According to [17-20] and our experimental tests, the hit ratio of (N-2)/(N-1) can be easily achieved.

According to (5), it can be concluded that in the worst case

$$h \propto t$$

It suggests that in real traffic, $t \ge (N-1)h + 1$ always holds true. This conclusion is in consistent with subsequent experimental results (see Figure 17).

IV. The New Incremental Update Mechanism

When an update message arrives, the routing table should be updated as fast as possible. Specifically, the whole update process is divided into three steps (see Figure 7): 1) trie update; 2) TCAM update; 3) DRed update. When the three steps are all finished, the update message takes into effect. Then how to evaluate the performance of incremental update? Time to Fresh (TTF) is defined in this paper, including TTF1 (TTF-trie), TTF2 (TTF-TCAM), and TTF3 (TTF-DRed).

The update involved in TCAM will interrupt routing lookup, resulting in decrease of lookup rate. As a result, TTF2 and TTF3 become more important than TTF1.

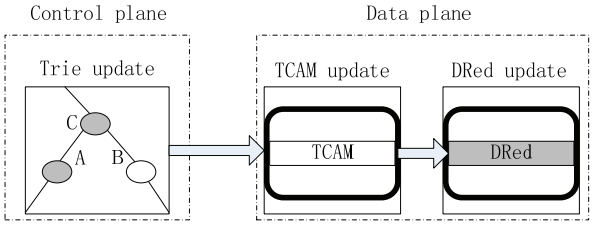


Figure 7. The whole incremental update process of CLUE.

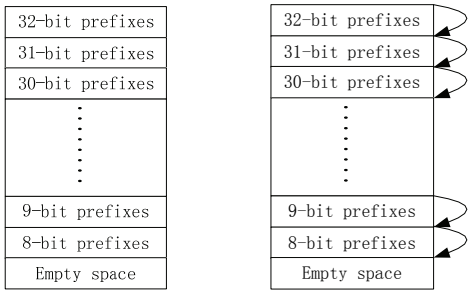
A. Trie Update

The performance of trie update is evaluated by TTF1 (TTF-trie), which means the average computing time of updating the trie. If no compression algorithm is adopted, TTF1 is minimal, and is regarded as ground-truth.

Because CLPL mechanism adopted no compression algorithm, thus its TTF1 is minimal, and TTF1-CLPL is defined to represent its TTF-trie. To be different, we adopt previously proposed ONRTC algorithm to compress the trie, and the update time is represented by TTF1-CLUE. The process of incremental update always keeps the trie non-overlap. Experimental results show that TTF1-CLUE is a little bit longer than TTF1-CLPL.

B. TCAM Update

Generally speaking, in order to reduce domino effect, the common method is to keep some redundancy at the end of TCAM. A naive solution [21] is shown in Figure 8(a). When a prefix is inserted, it will move all the following prefixes one by one, thus has a time complexity *O(n)* in the worst case, which is clearly undesirable.



(a) a naive solution (b) a classical solution

Figure 8. Two approaches to reduce domino effect

A sophisticated solution [21] is based on the observation that two prefixes with the same length can be placed interchangeably. This suggests that the domino effect can be reduced. As shown in Figure 8 (b), there is only a partial ordering constraint among all prefixes. This approach enables an empty memory location to be found in at most 32 prefix shifts. This is a classical method without extra overhead, while others need additional cost. Thus we assume it is adopted in CLPL, and is selected to be compared with

CLUE. Experimental results show that this solution needs 14.994 shifts in average.

The previous work [1, 8] of our group didn't emphasize this too much. After overlap is eliminated, the updating method of TCAM becomes clear and simple: when inserting a prefix, just write it to the end of TCAM; when deleting a prefix, just cut the last prefix to replace it. Therefore, CLUE needs one shift at most to handle an update message.

The performance of TCAM update is evaluated by TTF2 (TTF-TCAM), which indicates the update time of TCAM. Whatever partition algorithms are adopted, TCAM update must wait till the trie update is finished. The current partition algorithms probably need to change more than one prefix when one update message arrives.

It is obvious that our new TCAM update mechanism is much more efficient than all the traditional TCAM update mechanisms. This conclusion is consistent with the subsequent experimental results.

C. DRed Update

After the update of TCAM, in order to guarantee synchronization and correctness, the DRed must be updated, too. As mentioned above, CLPL adopts RRC-ME algorithm and its update algorithm.

When inserting or deleting a prefix in home TCAM, RRC-ME algorithm must look up the trie and find all the prefixes may be changed by this update. During this process, SRAM must be visited several times, which is a waste of time.

In contrast, when inserting a prefix in home TCAM, CLUE's DRed needs no change; when deleting a prefix, CLUE just lookup it in the DRed. If it exists, then just delete it; otherwise, do nothing.

It is clear that CLUE is much more efficient than CLPL in DRed update. This conclusion is also in accord with the subsequent experimental results.

TTF is the sum of TTF1, TTF2, and TTF3. The whole TTF comparison between CLPL and CLUE is given in the subsequent experimental results.

V. EXPERIMENTAL RESULT

A. Experimental Settings

1) Trace

TABLE I. LOCATIONS OF ROUTERS.

ID	Location	ID	Location
rrc01	LINX, London	rrc11	New York (NY), USA
rrc03	AMS-IX, Amsterdam	rrc12	Frankfurt, Germany
rrc04	CIXP, Geneva	rrc13	Moscow, Russia
rrc05	VIX, Vienna	rrc14	Palo Alto, USA
rrc06	Otemachi, Japan	rrc15	Sao Paulo, Brazil
rrc07	Stockholm, Sweden	rrc16	Miami, USA

The RIB packets are taken from www.ripe.net [21] at RIPE NCC, Amsterdam, which collects default free routing updates from peers. In order to objectively test the performance of ONRTC algorithms, the RIB packets at 8:00 on October 1 in 2011 from 12 routers are selected. (There are 16 routing tables available in www.ripe.net, but four of them don't update to present). Table III shows the routers' location.

In the routing update experiment, two traces are selected. In order to measure TTF-ratio, the update data from 2011.10.01/08:00 to 2011.10.02/08:00 is selected.

With regard to real traffic, trace from [23] is selected. The traffic from 20:59 to 21:14 on 2011.02.17 in Chicago is downloaded and parsed.

Our lab owns a TCAM (CYNSE70256). It supports 256K entries with 36-bit width. It can operate at a speed of up to 41.5 MHz by looking up 36 bit-width entries. It suggests that each lookup costs

$$1s/41.5\text{MHz} \approx 24\text{ns}$$

Generally speaking, the update time of each lookup is roughly equal to time of moving a prefix. Therefore, 24ns is regarded as the time cost of moving one prefix in TCAM.

2) Computer Configuration

Our experiments are carried out on a windows XP sp3 machine with Pentium (R) Dual-Core CPU 5500@2.80GHz and 4G Memory.

B. Experiments on Compression by ONRTC

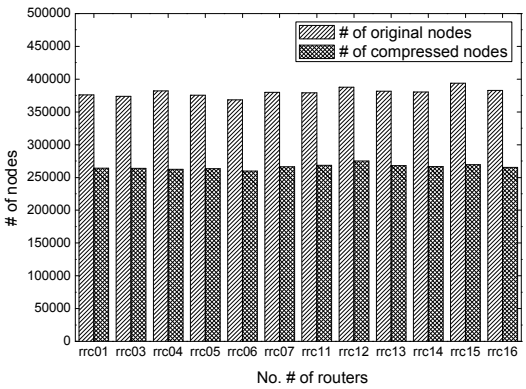


Figure 9. FIB size before and after compression on 12 routers.

The ONRTC compression results of 12 routers are shown in Figure 9. The taller bars are the original FIB size, while the lower are the FIB size after compression by ONRTC algorithm. According to the results, the compressed prefix number is 71% of the original in average, and the compression time is around 39 milliseconds.

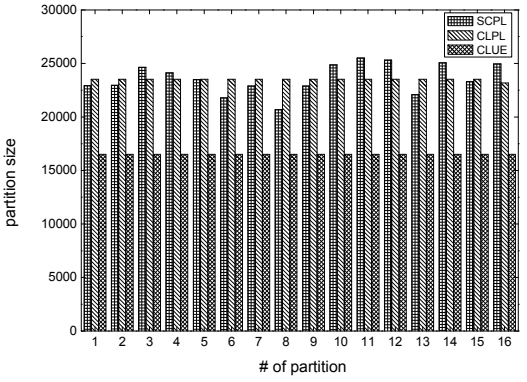


Figure 10. partition comparison among the three algorithms.

Figure 10 shows the partition results of three algorithms: SCPL algorithm, CLPL algorithm, and CLUE algorithm. The same experiments are conducted on 12 routers, with only one shown in the figure, because the results are similar. As shown in the figure, SCPL cannot split prefixes evenly, and CLPL split prefixes evenly at the cost of redundancy. In contrast, CLUE splits prefixes evenly with no redundancy, with much fewer prefixes in one bucket than both SCPL and CLPL. Besides, as the number of partitions rises, SCPL and CLPL introduces more redundancy (see Figure 6 in [1]), while CLUE still has no redundancy.

C. *Experiments on TTF*

The x-axis of Figure 11~15 stands for the arrival time of update messages. For example, 201010231945 means 2010.10.10/23:19:45.

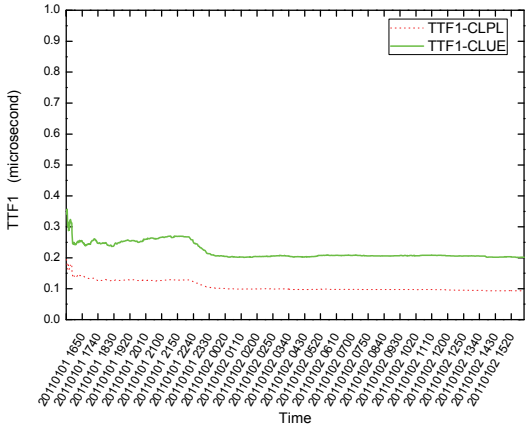


Figure 11. TTF1 comparison between CLPL and CLUE.

Figure 11 shows TTF1 (TTF-trie) of CLUE (ONRTC) and CLPL (ground-truth). It can be observed that TTF1 of CLUE is a little taller than ground-truth. TTF1 of CLUE ranges from 0.1924 microseconds to 0.3574 microseconds with a mean of 0.2210 microseconds. Because TTF1 doesn't interrupt routing lookup, a litter bigger TTF1 of CLUE doesn't influence system performance.

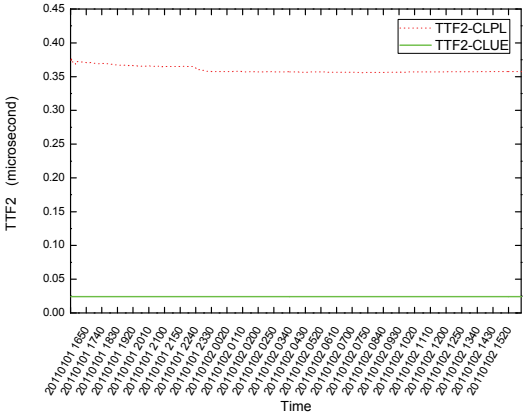


Figure 12. TTF2 comparison between CLPL and CLUE.

Figure 12 shows TTF2 (TTF-TCAM) of CLUE and general method (see Figure 8(b)). As mentioned in experimental settings, 24ns is regarded as the time cost of moving one prefix in TCAM. TTF2 of CLPL ranges from 0.3558 microseconds to 0.3782 microseconds with a mean of 0.3598 microseconds. In contrast, as mentioned above, CLUE needs only one shift (*O(1)*) to handle an update message, which means 0.024 microseconds for each update.

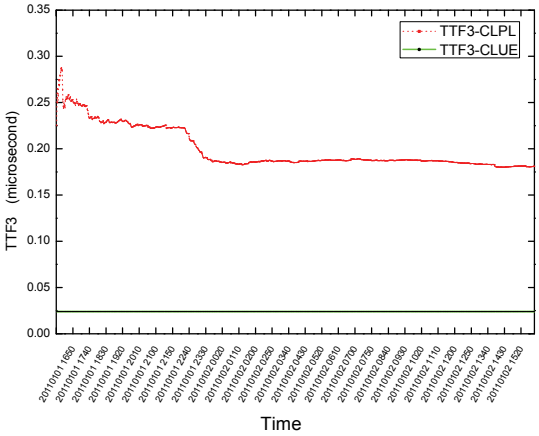


Figure 13. TTF3 comparison between CLPL and CLUE.

To evaluate the TTF3, we plot TTF-DRed in Figure 13. TTF3 of CLUE still maintains 0.024 microseconds; while TTF3 of CLPL ranges from 0.1802 microseconds to 0.2878 microseconds with a mean of 0.1993 microseconds. In other words, TTF3 of CLPL is 8.3 times of that of CLUE in average, and 11.99 times in worst case.

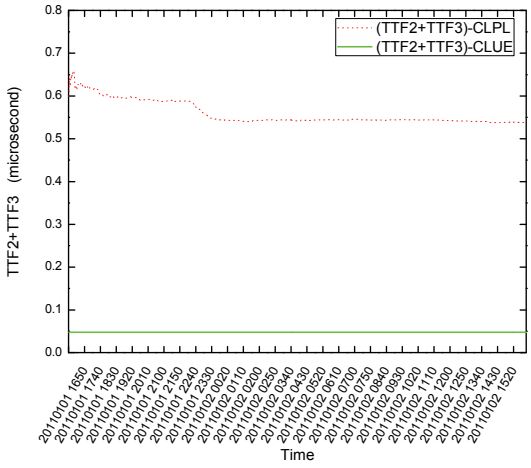


Figure 14. TTF2+TTF3 comparison between CLPL and CLUE.

As aforementioned, TTF2 and TTF3 are more important than TTF1, because TTF1 is the time cost in the control plane which doesn't interrupt routing lookup. In other words, both TTF2 and TTF3 influence the system performance. Therefore, the comparison of TTF2+TTF3 between CLPL and CLUE is shown in Figure 14. Results show that TTF2+TTF3 of CLUE is 4.29% of CLPL in average and 3.65% in worst case.

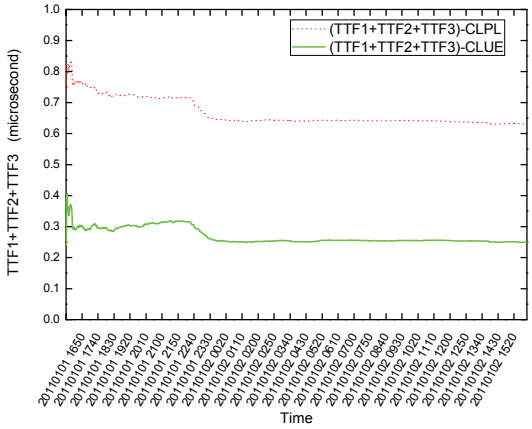


Figure 15. TTF1+TTF2+TT3 comparison between CLPL and CLUE

TTF, which is the sum of TTF1, TTF2, and TTF3, measures a router's sensitivity to the changes of the network state. Figure 15 shows the TTF of CLPL and CLUE. In the figure, the TTF of CLPL ranges from 0.6303 microseconds to 0.8342 microseconds with a mean of 0.6664 microseconds. In contrast, TTF of CLUE is only 0.2690 microseconds in average. In other words, TTF of CLPL is 234% of that of CLUE.

D. Experiments on Parallel Lookup

TABLE II. WORKLOAD ON DIFFERENT PARTITIONS AND TCAM CHIPS.

# of TCAM chips	#of Bucket	Range Low	Range High	Percent of partition	Percent of TCAM
1	2	38.103.176.0	61.91.89.255	21.92%	77.88%
	12	97.69.128.0	119.46.79.255	10.57%	
	20	194.133.118.0	196.11.124.255	9.18%	
	…				
2	23	202.30.78.0	203.128.191.255	4.52%	17.43%
	31	216.207.89.0	255.255.255.255	3.32%	
	8	72.9.88.0	77.79.211.255	3.13%	
	…				
3	16	168.87.144.0	183.87.78.255	0.81%	4.54%
	13	119.46.80.0	134.75.216.255	0.72%	
	5	65.68.16.0	66.133.181.255	0.70%	
	…				
4	11	91.209.9.0	97.69.127.255	0.08%	0.16%
	10	85.95.88.0	91.209.8.255	0.07%	
	0	0.0.0.0	12.177.231.255	0.00%	
	…				

As shown in TABLE II, routing table from rrc01 is split into 32 partitions evenly by CLUE. After test by real traffic, the workload of each partition is shown in Column 'Percent of partition'. It can be observed that workload among different partitions varies a lot. To simulate bursty traffic, the partitions are sorted by the workload percentage in descending order. The first 8 partitions are mapped to $TCAM_1$, while the second, third and the fourth 8 partitions are mapped to TCAM 2, 3, 4, respectively. This is a possible mapping situation with extremely uneven workload among TCAMs.

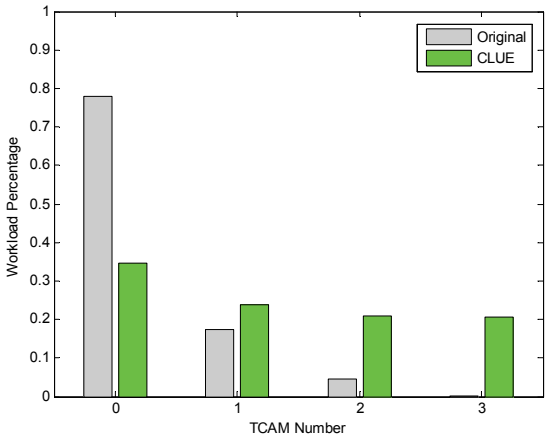


Figure 16. Load balance of workload distribution by CLUE.

The grey bars labeled 'Original' in Figure 16 show the extremely uneven workload distribution of Table II. An experiment using this distribution is designed to evaluate the function of CLUE's workload balancing. In the simulation process, each TCAM takes 4 clocks to process a packet, while a packet arrives per clock. The FIFO is set to 256 and redundancy size is set to 1024 prefixes. The green bars show the traffic distribution balanced by CLUE. It can be seen that the green bars labeled 'CLUE' are much more even than 'Original'. It can be concluded that CLUE can achieve excellent workload balancing even in the worst case.

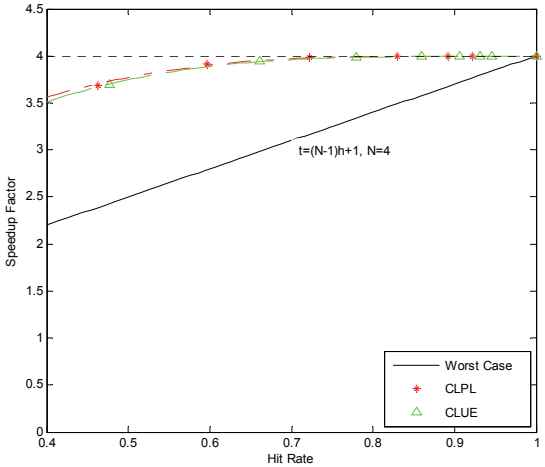


Figure 17. Speedup factor comparison between CLPL and CLUE and the worst case

Figure 17 shows the relationship between Hit Rate and Speedup Factor. It is a comparison among CLPL, CLUE, and the worst case in theory. The dotted lines of CLPL and CLUE are the results of cubic curve fitting. Both CLPL and CLUE are much better than the worst case, which is consistent with previous theory results. This figure suggests that the speedup factor rises as hit rate rises. In terms of

CLPL and CLUE, the same Speedup Factor will be achieved by the same hit rate, because they almost overlap.

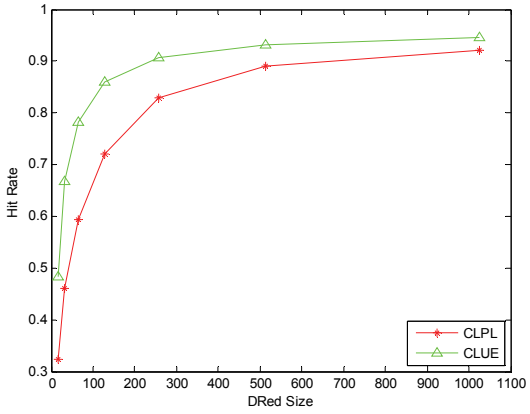


Figure 18. Hit rate comparison between CLPL and CLUE.

The relationship between DRed Size and Hit Rate is plotted in Figure 18. The top curve is the result of CLUE, and the other one belongs to CLPL. It indicates that CLUE achieves much higher Hit Rate than CLPL with the same DRed Size. Whereas Figure 17 shows Hit Rate determines Speedup Factor, then it can be indirectly concluded that CLUE achieves much higher Speedup Factor than CLPL with the same DRed Size.

VI. Discussion and Conclusion

Due to the explosive increase of Internet volume and traffic, routing tables in backbone routers have been increasing approximately 15% in size annually. Meanwhile, the link transmission speed of backbone routers has increased to gigabit-per-second. Consequently, the backbone routers are facing CLUE: routing table Compression, fast routing Lookup, and fast incremental UpdatE.

Traditional algorithms seldom cover the three problems simultaneously. Therefore, we propose CLUE. The design philosophy of CLUE is that we should not view the three problems isolatedly and statically, avoiding one-sidedness.

First, CLUE adopts ONRTC algorithm, which supports parallel lookup and fast incremental update.

Second, several improvements are made based on CLPL mechanism, achieving lower hardware cost.

Third, TTF is firstly defined to describe the sensitivity of a router, including TTF-trie, TTF-TCAM, and TTF-DRed.

Extensive experimental results show that, compared with CLPL, CLUE needs much less hardware resource and shorter update time to achieve the same speedup factor.

Next, we will continue our work in the following areas: 1) we are applying CLUE on IPv6 routing tables; 2) we are applying our algorithms and testing its actual performance in real routers.

References

[1] Lin, D., Zhang, Y., Hu, C., Liu, B., Zhang, X., Pao, D. Route Table Partitioning and Load Balancing for Parallel Searching with TCAMs. In Proc. IPDPS, 2007.

[2] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 Addre-ss Allocation and the BGP Routing Table. ACM SIGCOMM Computer Communication Review, vol. 35, pp. 71–80, January 2005.

[3] R. Draves, C. King, S. Venkatachary, and B. D. Zill. Constructing Optimal IP Routing Tables. In Proc. IEEE INFOCOM, 1999.

[4] X. Zhao, Y. Liu, L. Wang, and B. Zhang. On the Aggregatability of Router Forwarding Tables. In Proc. IEEE INFOCOM, 2010.

[5] Heeyeol Yu. A memory- and time-efficient on-chip TCAM minimizer for IP lookup. DATE '10 Proceedings of the Conference on Design, Automation and Test in Europe, 2010.

[6] Qing Li, Dan Wangy, Mingwei Xu, Jiahai Yang. On the Scalability of Router Forwarding Tables. Nexthop-Selectable FIB Aggregation. In Proc. IEEE INFOCOM, 2011.

[7] F. Zane, G. Narlikar, A. Basu. CoolCAMs: Power-Efficient TCAMs for Forwarding Engines. In Proc. INFOCOM, 2003.

[8] Zheng, K., Hu, C., Lu, H., Liu, B. A TCAM-based distributed parallel IP lookup scheme and performance analysis. IEEE/ACM Trans. Netw. 14, 863–875, 2006.

[9] Mohammad J. Akhbarizadeh and Mehrdad Nourani. Efficient Prefix Cache for Network Processors, High Performance Interconnects 2004, pp.41-46, August 2004.

[10] Bin Zhang, Jiahai Yang, Jianping Wu, Qi Li, Donghong Qin. An Efficient Parallel TCAM Scheme for the Forwarding Engine of the Next-generation Router. In Proc. IFIP/IEEE IM, 2011.

[11] V. Srinivasan and G. Varghese. Fast IP lookups using controlled prefix expansion, ACM TOCS, vol. 17, pp. 1–40, Feb. 1999.

[12] R. Panigrahy, S. Sharma. Reducing TCAM Power Consumption and Increasing Throughput. Proceedings of HotI 2002, pp.107-112, August 2002.

[13] Abilene. http://www.abilene.iu.edu/noc.html.

[14] E. Ng and G. Lee. Eliminating sorting in ip lookup devices using partitioned table. In The 16th IEEE International Conf. on Application Specific Systems, Architecture and Processors, 2005.

[15] K. Jinsoo and K. Junghwan. An efficient ip lookup architecture with fast update using single-match tcams. In WWIC, 2008.

[16] Bin Zhang, Jiahai Yang, Jianping Wu, Qi Li, Donghong Qin. An Efficient Parallel TCAM Scheme for the Forwarding Engine of the Next-generation Router. In Proc. IFIP/IEEE IM, 2011.

[17] Woei-Luen Shyu, Cheng-Shong Wu, and Ting-Chao Hou. Efficiency Analyses on Routing Cache Replacement Algorithms, ICC'2002, Vol.4, pp.2[24]2-2236, April/May 2002.

[18] Tzi-cker Chiueh,Prashant Pradhan. High-Performance IP Routing table Lookup Using CPU Caching. In Proc. INFOCOM, 2003.

[19] Bryan Talbot, Timothy Sherwood, Bill Lin. IP Caching for Terabit Speed Routers. In Proc. GLOBECOM, 1999.

[20] Mohammad J, Akhbarizadeh and Mehrdad Nourani. Efficient Prefix Cache for Network Processors. High Performance Interconnects 2004, pp.41-46, August 2004.

[21] D. Shah and P. Gupta, Fast incremental updates on ternary-CAMs for routing lookups and packet classification. In Proc. Hot Interconnects 8,Aug. 2000, pp. 145–153.

[22] RIPE Network Coordination Centre.
http://www.ripe.net/data-tools/stats/ris/ris-raw-data.

[23] The CAIDA Anonymized 2011 Internet Traces - <20110217> Colby Walsworth, Emile Aben, kc claffy, Dan Andersen, http://www.caida.org/data/passive/passive_2009_dataset.xml

[24] Routing Table Compression and Update Website.
http://s-router.cs.tsinghua.edu.cn/~yangtong/