TAR: Traffic Adaptive IPv6 Routing Lookup Scheme

Xinyi Zhang[1,2], Zhiyuan Xu[2], Huaiyi Zhao[2,3], Yanbiao Li[1,2], Gaogang Xie[1,2]
[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

ABSTRACT

IP lookup aims at identifying the longest prefix match within routing tables to determine the forwarding path for packets. The increase in IPv6 address length makes IP lookup particularly difficult, requiring more efficient lookup mechanisms to handle the expanded address space and ensure the timely and accurate routing of packets. Existing IPv6 lookup algorithms focus on constructing efficient data structures to enhance lookup performance. However, given the inherent characteristics of network traffic distribution and the varying hit probabilities of nodes within the lookup structure, there is still room for optimization. Our study introduces a new IPv6 routing lookup scheme that utilizes the characteristics of network traffic to guide the construction of lookup data structures. Experimental results indicate that the lookup performance of the traffic-adaptive algorithm is 1.1 to 2.2 times that of conventional algorithms. Additionally, our work designs a traffic-adaptive routing system, which includes an adaptive cache structure capable of responding to dynamic changes in network traffic. The throughput of our proposed system is 1.7 to 2.8 times that of systems implementing only basic algorithms.

CCS CONCEPTS

• **Networks → Data path algorithms**.

KEYWORDS

IPv6 Route Lookup, Traffic Adaptive Approach

1 INTRODUCTION

IP lookup[6, 14, 16] is crucial not only for core routers but also for software-based switches and routers within Network Function Virtualization (NFV), making it a longstanding focus and challenge in research. IP lookup process principally involves inspecting the destination IP address of incoming packets to identify the longest prefix match within a router's Forwarding Information Base (FIB). The outcome of this match dictates the packet's forwarding path and method, guaranteeing its efficient navigation across the Internet to its intended destination. With the widespread adoption of IPv6 in recent years, the necessity for effective lookup mechanisms to manage the expanded addressing space has introduced unprecedented challenges for IP lookup technology.

Existing IPv6 lookup technologies can broadly be classified into two categories: software-based schemes[2, 6, 8, 17] and hardware-based schemes[4, 5, 12]. Software-based IPv6 lookup algorithms are implemented within the router's operating system, utilizing the general-purpose CPU for the lookup process, thereby offering high flexibility. Most existing optimization techniques exploit cache architectures and memory access patterns to optimize data structures and algorithms, thereby enhancing performance. On the other hand, hardware-based solutions employ specialized hardware components, such as Ternary Content Addressable Memory (TCAM), Field-Programmable Gate Arrays (FPGAs), or Application-Specific Integrated Circuits (ASICs), all designed specifically for high-speed lookup operations. These solutions are generally faster but offer less flexibility compared to software-based implementations.

The process of IPv6 packet lookup primarily involves two key components: "traffic" and "rule set." Traffic refers to the actual packets moving through the network, whereas rule set (FIB) defines the rules for processing and forwarding packets based on their destination IPv6 addresses. Evaluating the performance of the traffic against the rule set directly reflects the efficiency of an IPv6 lookup algorithm. While most IPv6 lookup algorithms[1, 2, 5, 6, 17] have considered the distribution of rules in their structural design and implementation, they overlook the characteristics of the network traffic. Therefore, if it's possible to leverage the characteristics of network traffic to guide the construction of lookup structures and reduce the memory access overhead of popular traffic, there is the potential to enhance lookup performance.

In this paper, we propose a traffic-adaptive IPv6 routing lookup algorithm and system. The principal contributions of our work can be summarized as follows.

(1) First, we introduce the idea of utilizing traffic information to build data structures specifically for IPv6 lookup algorithms. We have developed several traffic-adaptive implementations, which are collectively termed the TA-tree. The data structures constructed based on traffic have been experimentally tested and shown to significantly improve lookup performance while maintaining feasible storage overhead and rapid construction speed.
(2) Second, we have developed a Traffic Adaptive Routing (TAR) system capable of adapting to the dynamic changes in network traffic. Considering the bursty and dynamic nature of network traffic, our system integrates *monitoring module* and *lookup module* to facilitate the dynamic update of lookup structures and real-time decision-making for lookup procedure. This adaptive approach not only addresses the inherent variability of network traffic patterns but also ensures high-performance lookup operations under diverse network conditions. The specific implementation details will be introduced in the following sections.
(3) Third, the traffic-adaptive IPv6 routing lookup algorithm and system represents the first systematic approach to leveraging traffic characteristics for guiding the construction of IP lookup data structures. This innovative idea moves beyond the traditional focus on optimizing data structures themselves. It holds significant importance for enabling intelligent awareness and transmission within the network data plane.

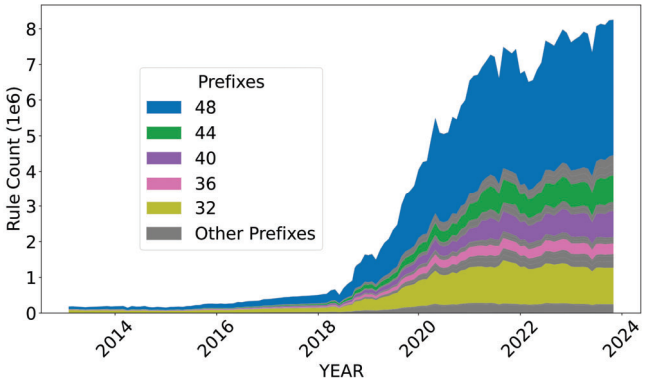


Figure 1: Area stack plot of IPv6 prefix length counts

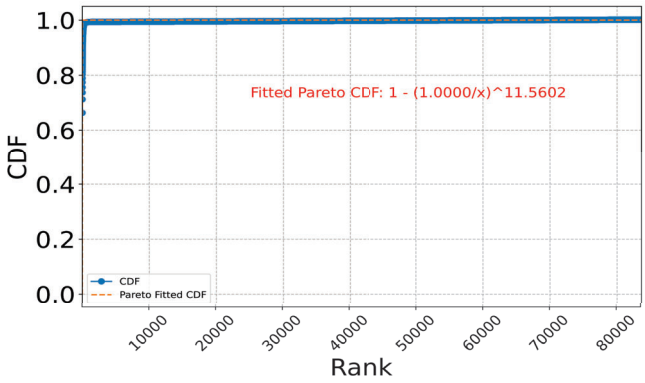


Figure 2: CDF of IPv6 traffic with Pareto Fitted Curve

In subsequent sections of the paper, Section 2 primarily covers the research background and motivation. Section 3 details the design of the algorithm and system implementation, while Section 4 presents the experimental design and test results. Finally, the paper concludes with a summary.

2 BACKGROUND AND MOTIVATION

To delve into the characteristics of IPv6 prefix length distribution, we analyze open-source datasets from RIPE[10]. We compile statistics on the distribution of lengths of destination IPv6 prefixes in the RRC00 datasets that cover the first day of each month from January 2013 to November 2023, resulting in an area stack plot, as illustrated in Figure 1. This plot illustrates the changes in rule counts for different network prefix lengths over time. Each layer in the plot represents a specific prefix length, with its thickness indicating the rule count for that prefix length on a given date. We have observed a sharp increase in the total number of prefixes over the past decade. The combined count of prefix lengths '48', '32', '44', and '40' exceeds 77% in November 2023. Most existing IPv6-based optimization solutions are largely based on this observation, guiding the construction of data structures with prioritized handling for prefix lengths that encompass a larger number of rules.

In addition to analyzing the characteristics of rule sets, the study of IPv6 traffic distribution is equally important. We utilize authentic

Table 1: A Sample IPv6 Ruleset

Rule #	Prefix	Next Hop
A	0*	1
B	01000*	2
C	0111*	3
D	10*	4
E	11*	5

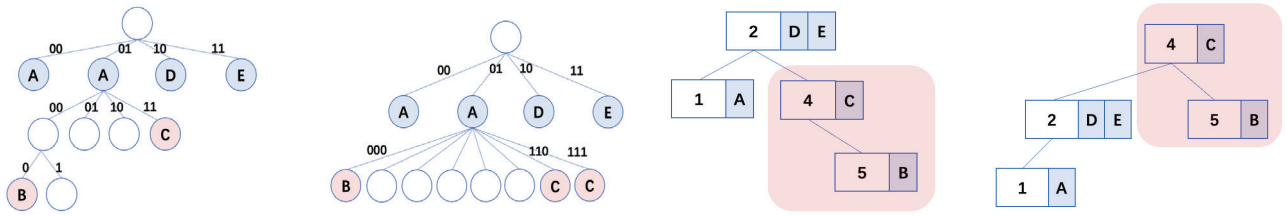
traffic data obtained from the MAWI [9](Measurement and Analysis on the WIDE Internet). Taking the dataset from 2022/04/13 at 20:30 (includes traffic over a duration of 15 minutes) as an example, we conduct a statistical analysis of destination IPv6 addresses in packets, measuring the frequency of each address. After ranking these addresses by frequency, we generate Cumulative Distribution Functions (CDFs) and fitting plots, as shown in Figure 2. The analysis reveals that the distribution of IPv6 traffic is heavy-tailed and skewed, with a small number of addresses generating the majority of traffic. This pattern aligns with the "Prefix-related characteristics of IPv6 traces" discussed in literature [7].

In IPv6 lookup structure design, the primary methods for IP lookup are categorized into Trie-based and BST-based approaches. Trie-based methods[1–3, 17] utilize trie structures to incrementally narrow down the search for an IP address. Each node in the trie represents a segment of the IP address, enabling efficient pinpointing of the target entry. The rules corresponding to Table 1 are represented through a trie as shown in Figure 3(a). BST-based methods [6, 8, 15] store IP prefixes with different length into different hash tables. These hash tables are then organized into an asymmetric Binary Search Tree, as illustrated in Figure 3(c).

However, analysis of real-world traffic demonstrates that traffic distribution exhibits locality. We assume that rules B and C are the two most frequently hit rules in Table 1, as indicated by the red part in Figure 3. For Trie-based structure, constructing a multibit trie as shown in Figure 3(a) would require at least four memory accesses for packets corresponding to rule B. In contrast, if considering traffic characteristics and placing B and C at the same leaf node level, the number of memory accesses for packets corresponding to rule B could be reduced by one, as shown in Figure 3(b). For the BST-based structure shown in Figure 3(c), if the lookup structure is built based on the number of rules corresponding to each hash table, packets corresponding to rule B would require at least three memory accesses. Conversely, by taking into account the distribution of packet hits, constructing a lookup structure as shown in Figure 3(d) can reduce the memory access count for packets corresponding to rules B and C by one each. Therefore, by designing IPv6 lookup structures based on traffic patterns, we can significantly improve the IPv6 lookup efficiency.

3 TRAFFIC ADAPTIVE SCHEME

In this section, we first introduce the construction of the traffic adaptive data structures for both trie-based and BST-based schemes. After that, we introduce the design of the Traffic Adaptive Routing system (TAR).



(a) Multibit trie (b) Traffic adaptive multibit trie (c) BST (d) Traffic adaptive BST

Figure 3: Data structure constructed based on a sample ruleset

3.1 Algorithm Design

Given that the IP lookup mechanism mainly utilizes tries and binary search trees, this section designs traffic adaptive implementations for each type of algorithm and collectively refers to such data structures as TA-Tree.

Trie-based: In trie-based IPv6 lookup methods, each node N represents a decision point, potentially encompassing information across multiple bits. During IPv6 prefix lookup, traversal begins at the root, with several bits of the address inspected at each node to determine the next descendant node or to retrieve the associated next-hop information. The *stride* denotes the number of bits examined concurrently at a node, striking a balance between search efficiency and memory utilization. Typically, nodes with larger strides examine more bits simultaneously, facilitating faster lookups at the expense of increased memory consumption due to a higher number of child nodes. While previous studies primarily focus on efficient data structures to store child pointers and next-hop information, prefix expansion techniques[11], and compression mechanisms, they overlook traffic characteristics during trie construction. In our approach, we integrate traffic density data to tailor the trie structure to specific traffic patterns, enhancing its optimization.

Intuitively, for nodes with higher traffic density, increasing the stride can enhance lookup efficiency, while for nodes with less traffic, reducing the stride should conserve memory. Although a larger stride generally reduces the number of memory accesses, it exponentially increases memory consumption while only linearly decreasing the number of memory accesses. Therefore, globally increasing the stride is not always practical. Instead, incorporating traffic characteristics to dynamically adjust the stride based on traffic patterns is more effective.

Formally, in the context of trie-based method, R denotes an IPv6 rule set, and T represents a specific trie. Each node N in T is associated with $R(N)$ rules, and $S(N)$ indicates its stride. When a packet arrives at node N, s is a child of N given the stride $S(N)$, $F(s)$ represents the frequency of visiting node s, and $E(A(N))$ denotes the expected number of subsequent memory accesses. $M(N, S(N), R(N))$ denotes the memory usage for storing node N's information when the stride is $S(N)$, and $M(T)$ represents the total memory usage of the trie. Under the constraint that the total memory is within the capacity $M_{\max}$, we seek to determine the configuration c of strides $S(N)$ in trie T (i.e., $c = [S(N_1), S(N_2), \cdots]$) that minimizes $E(A(N_{\text{root of } T}))$, which can be calculated by the recursion below:

$$E(A(N)) = \begin{cases} 0 & R(N) = 1 \\ \sum_{s \in sons(N, S(N))} E(A(s)) \cdot F(s) + 1 & \text{otherwise} \end{cases}$$

The mathematical formulation of the problem is as follows:

$$\begin{aligned} \arg\min_{c} \quad & E(A(N_{\text{root of } T})) \\ s.t. \quad & M(T) = \sum_{N \in T} M(N, S(N), R(N)) \leq M_{\max} \end{aligned}$$

BST-based: In BST-based IPv6 lookup scheme, IPv6 address are typically categorized into subsets based on their prefix lengths, with distinct hash tables managed for each subset. These hash tables are then stored in separate nodes of a binary search tree (BST), facilitating binary search on prefix lengths during lookups. Traditional BST-based algorithms tend to position hash tables with more rules closer to the root, as they have a higher probability of rule matching upon probing, which might lead to earlier termination of the search process and thus reduce the average search time. However, rule matching probability is influenced not only by rule counts but also by traffic distribution.

In the construction of BST-based TA-Tree, a new metric termed "popularity" ($P(N)$) is proposed to assess the likelihood of matching at each node. This metric considers both the number of rules associated with each prefix length and the hit counts for these rules based on historical traffic patterns.

$$P(N) = \alpha \cdot \frac{R(N)}{\sum_{N \in Nodes} R(N)} + \beta \cdot F(N),$$

Here, $R(N)$ denotes the number of rules at node N, $F(N)$ represents the percentage of packets that finally find a match at node N in the historical traffic, and α and β are weights adjusting the impact of rule counts and hit counts, respectively. Nodes exhibiting higher popularity, indicative of higher chance of rule matches, are strategically positioned closer to the root, thereby enhancing search efficiency.

We have applied these strategies to trie-based methods DIR-24-8 and Poptrie, as well as BST-based method Asymmetric Binary Search Tree (ABST), to evaluate their effectiveness. Importantly, our approach does not alter the fundamental structure of tries or binary search trees, thus allowing potential application to other algorithms. Optimization techniques such as Rope[14] and tree

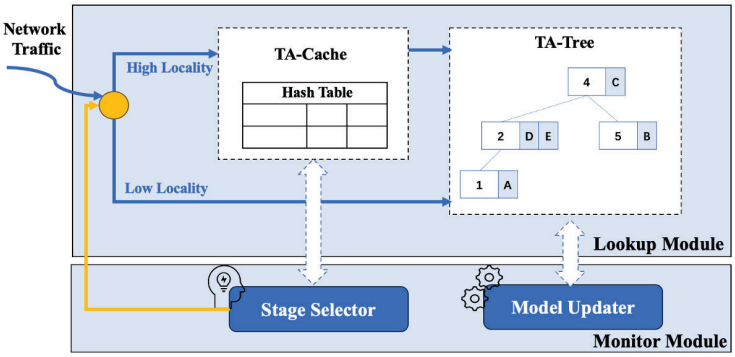


Figure 4: Overview of TAR system

rotation[6] remain applicable to our approach. Moreover, dynamic rule insertion and deletion are still supported if the original algorithm accommodates these operations.

3.2 System Implementation

Even though TA-Tree utilizes traffic characteristics to guide the construction of the efficient data structure for IPv6 lookup, considering the temporal locality and dynamic variability of network traffic, we further design a Traffic Adaptive Routing system (TAR). The TAR system contains a lookup module comprising *TA-Cache* and *TA-Tree*, and a monitoring module consisting of a *Stage Selector* and *Model Updater*, which are detailed below.

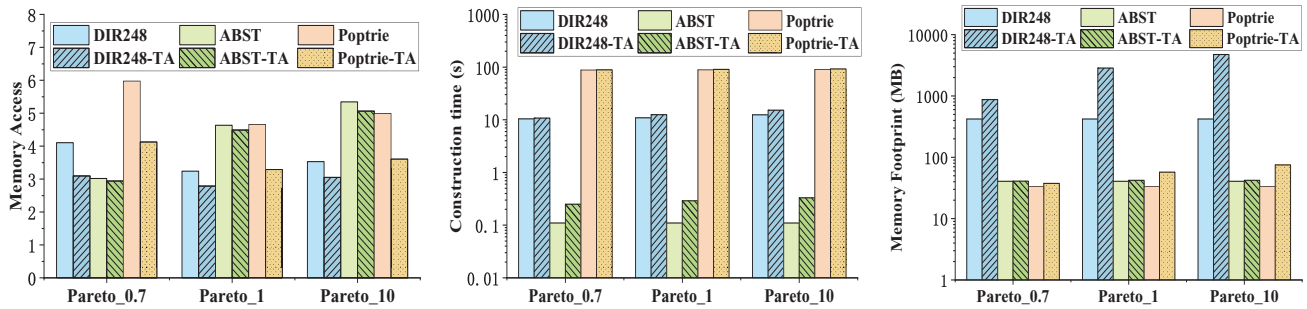
3.2.1 Lookup Module. In practice, packets with the same destination IP address are forwarded to the same next hop. To further accelerate the lookup rate, we construct a lightweight TA-Cache with a hash table, where each entry stores an IP address, the corresponding next hop and a counter. Upon packet arrival, the TA-Cache is queried first. If the corresponding address is found, it is processed directly; otherwise, it is forwarded to the TA-Tree for further inquiry.

To address cache table thrashing in scenarios with long-tail traffic distribution and the burstiness of network traffic, TA-Cache only stores specific data items that contribute to the majority of the traffic and promptly removes entries that are no longer popular. Specifically, by sampling a small portion of traffic (e.g., 1/1000), for each sampled packet, the IP address is hashed to find the corresponding position in the hash table. If the position is empty, a new entry is created with its counter set to 1. If the position is not empty and the IP address matches, the counter is increased. If the position is not empty and the IP address does not match, the counter is multiplied by a decay factor α (e.g., 0.99). When the counter falls below 1, the entry is evicted and replaced. This eviction strategy ensures that current significant visitors remain in the cache, as their counters will be frequently incremented, while old significant visitors are timely removed due to exponential decay of their counters. Regarding storage overhead, since each entry in the TA-Cache occupies only 24 bytes (IP address 16 bytes, next hop 4 bytes, counter 4 bytes), a typical hash table containing hundreds of entries will consume only a few kilobytes of memory.

3.2.2 Monitoring Module. Although the presence of TA-cache can significantly improve network performance when network traffic exhibits high temporal locality, its presence can lead to additional caching for each packet when locality is low. To address this issue, the Stage Selector periodically reads counter values with a window size set to t. High network traffic locality is inferred when the hit counts of the top M frequently accessed IP addresses—where M equals 20% of the hash table entry count—exceed 70% of the total hit counts across the hash table. Should these criteria not be met, it indicates low network traffic locality, prompting the Stage Selector to deactivate the TA-cache and reroute packet queries directly to the TA-Tree. However, network traffic is dynamic. When the TA-Cache deactivates, the monitor module mirrors the data plane traffic at intervals of time t. Similarly, When the hit counts of the top 20% M frequently accessed IP addresses exceed 70% of the total hit counts, the TA-Cache can be promptly reactivated.

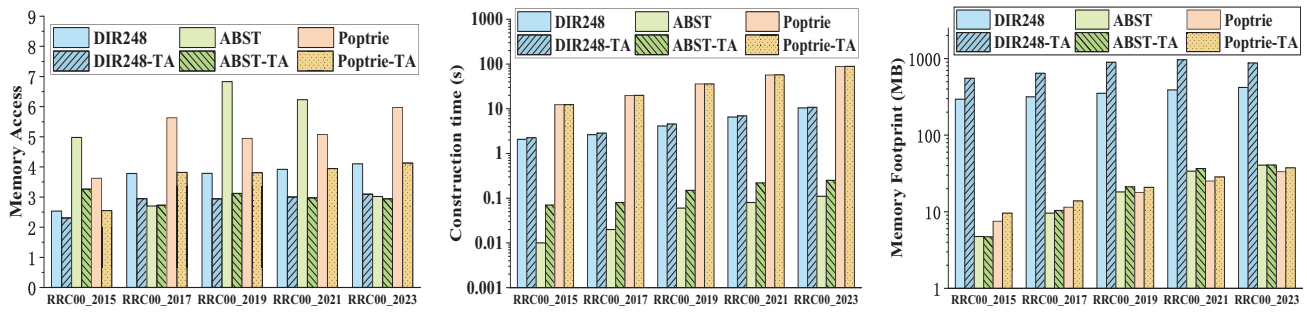
Additionally, considering that TA-Tree is built based on traffic patterns, the lookup structure must be updated promptly when the traffic distribution changes to maximize performance. To achieve this goal, we employ the following method: First, we compare the top N items in two time windows, t_2 and t_1, where N is defined as $0.1\% \times R(N)$ in our method. If more than half of the prefixes in the top N are inconsistent between t_2 and t_1, it indicates a need for structural reconstruction. The reconstruction process involves first establishing a new lookup structure B; once the new structure is completed, the pointer directing to the original lookup structure A is redirected to the new structure B, and the storage space of the original structure is released, thus performing an A/B swap. This A/B swapping strategy ensures the continuity of the lookup process and aligns the system with current network conditions, optimizing search efficiency. By doing so, the system can adapt to traffic fluctuations, thereby maintaining high performance continuously.

By integrating above strategies, our system not only dynamically responds to the evolving network environment but also establishes a logical framework for periodic data structure adjustments. This ensures the lookup system's efficiency and scalability, capable of handling the variability of network traffic patterns.



(a) memory access (b) construction time (c) memory footprint

Figure 5: Algorithm performance with different pareto b parameters (a=1)



(a) memory access (b) construction time (c) memory footprint

Figure 6: Algorithm performance with different datasets (pareto: a=1, b=0.7)

4 PERFORMANCE EVALUATION

4.1 Experimental Setup

Our evaluation is carried out on two servers connected by Mellanox ConnectX-5 100G NICs. Each server hosts dual Intel Xeon Gold 6338 CPUs with a 2.00GHz clock speed and is equipped with 256GB of RAM. These servers run Ubuntu 20.04.5, utilizing Linux kernel version 5.4.0.

In section 4.2, we implement traffic-adaptive versions of existing IP lookup algorithms, including DIR248-TA, Poptrie-TA, and ABST-TA, and conduct these evaluations on a dedicated server. We perform a comparative analysis of the proposed solution against existing solutions in terms of memory access, memory footprint, and construction time. In this section, only TA-Tree is evaluated.

In section 4.3, we utilize one server running the TAR system implemented with above algorithms with or without TA-Cache, and another running a traffic generator program based on DPDK to send traffic to the TAR system under test.

The ruleset for evaluation originates from RIPE[10] RRC00. We extract prefixes from the datasets at 8:00 AM on the first day of each year in 2015, 2017, 2019, 2021, and 2023 to form the ruleset for testing. The resulting rulesets contain 21809, 36767, 68668, 110973 and 174050 rules, respectively.

Considering traffic distribution impacts the performance of IP lookup schemes, we propose two methods to generate test traffic for a given ruleset.

Simulated Trace: We begin by selecting a prefix from the ruleset and randomly completing its remaining bits. Subsequently, we sample from Pareto distribution $P(x) = \frac{ab^a}{x^{a+1}}$ to determine how many times this address appears in the traffic[13]. This method grants us control over the traffic locality, enabling comparative evaluations of different algorithms across varying traffic scenarios. In Section 4.2, we use this method, fixing $a = 1$ and manipulating the value of b.

Real Traffic: Instead of sampling from a distribution, this method involves mapping each synthesized IP address to a corresponding address from a reference real traffic dataset (in our case, MAWI). The appearance frequency of each address is then replicated from the mapped address in the real traffic dataset. This approach preserves the locality of real traffic patterns, and is utilized in Section 4.3 to showcase the effectiveness of our algorithm in real-world scenarios. For security reasons, we could not find a real ruleset and corresponding real traffic from the same dataset. Therefore, this mapping is essential because directly using real traffic that does not align with the ruleset would result in the majority of packets being matched by the default rule, thereby reducing the experiment's relevance.

4.2 Algorithm Performance

4.2.1 Memory access. We first evaluate the average memory access frequencies of various algorithms. Initially, for the prefix rule set obtained by RR0CC 2023, we generate traffic based on different Pareto parameters b (where a lower value indicates higher locality) to examine the number of memory accesses under different algorithms, as depicted in Figure 5(a). Furthermore, we have selected several different rule sets to measure the memory access frequencies under a pareto parameter b of 0.7 shown in Figure 6(a). Our findings show that DIR248-TA, ABST-TA, and Poptrie-TA respectively improve performance by 1.1 to 1.3 times, 1.5 to 2.2 times, and 1.3 to 1.5 times compared to the original algorithms.

4.2.2 Construction time. The data structure construction results are shown in Figure 5(b) and Figure 6(b). The ABST-TA algorithm first constructs a tree for packet querying and then builds a traffic adaptive tree based on the hit counts of each hash table. This process doubles the construction time for solutions driven by traffic patterns. Conversely, in the DIR248-TA and Poptrie-TA algorithms, the construction relies solely on obtaining the access frequency at each node, eliminating the need for an additional IP lookup structure. Therefore, the construction efficiency of DIR248-TA and Poptrie-TA is comparable to that of basic algorithms.

Furthermore, among the studied datasets in MAWI, we did not observe any major change in traffic distribution that requires a complete reconstruction of the data structure within the 15-minute interval (default interval for MAWI traffic samples), except for the emergence and disappearance of bursty flows which can be handled by the TA-cache. This indicates that the construction time of the traffic adaptive algorithms is acceptable and TA-Trees can be timely updated to adapt to changes in network traffic.

4.2.3 Memory footprint. Regarding memory consumption, as depicted in Figure 5(c) and Figure 6(c), DIR248 inherently features a large stride (with default values set to 24 for the root node and 8 for subsequent nodes, as implied by its name), leading to higher storage overhead compared to other algorithms. The storage overhead for DIR248-TA sees an increase ranging from 50% to 158%. However, since the original DIR248 algorithm already consumes a substantial amount of memory that exceeds CPU cache capacity, and the additional increase remains within typical DRAM size and is thus deemed acceptable. Compared to ABST, the increase in storage cost for ABST-TA ranges from 0.2% to 17%. Meanwhile, the optimized solution for Poptrie incurs a 12% to 27% increase in storage overhead compared to Poptrie. Given the enhancements in performance, such modest increases in storage requirements are considered acceptable.

4.3 System Evaluation

We have implemented various algorithms and their corresponding traffic-adaptive versions on a DPDK-based system. For throughput evaluation, we utilize MAWI 202204140014 dataset to generate traffic on RRC00 rulesets. The evaluation results highlight a notable enhancement in system performance with the introduction of our traffic adaptive optimization. Furthermore, in real-world scenarios where temporal traffic locality is prevalent, the addition of the TA-cache (denoted by the suffix *WC*) substantially boosted lookup performance. Test results indicate that the throughput of systems based on DIR248-TA-WC, ABST-TA-WC, and Poptrie-TA-WC is 1.8 to 2.0 times, 1.9 to 2.8 times, and 1.7 to 2.4 times that of systems using only basic algorithms.

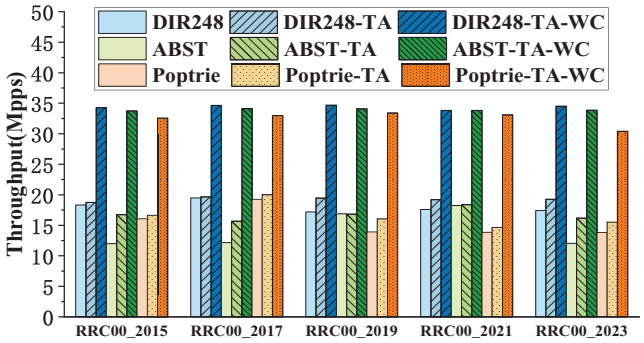


Figure 7: System evaluation with DPDK

5 CONCLUSION

In this paper, we propose a novel IPv6 lookup scheme designed to adapt to varying traffic patterns through the construction of specialized data structures. Experimental results demonstrate that our proposed TA-tree and TAR system significantly improve lookup performance compared to existing solutions. We believe that utilizing traffic characteristics to guide the construction of data structures will bring new insights into traditional IP lookup schemes.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their thoughtful suggestions. This work was supported by the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 62202447) and National Natural Science Foundation of China (Grant No. 62072430).

REFERENCES

[1] Hirochika Asai and Yasuhiro Ohara. 2015. Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 57–70.

[2] Hao Chen, Yuan Yang, Mingwei Xu, Yuxuan Zhang, and Chenyi Liu. 2022. Neurotrie: Deep Reinforcement Learning-based Fast Software IPv6 Lookup. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 917–927.

[3] P. Gupta, S. Lin, and N. McKeown. 1998. Routing lookups in hardware at memory access speeds. In *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98CH36169)*, Vol. 3. IEEE, San Francisco, CA, USA, 1240–1247. https://doi.org/10.1109/INFCOM.1998.662938

[4] Yi-Mao Hsiao, Ming-Jen Chen, Yu-Jen Hsiao, Hui-Kai Su, and Yuan-Sun Chu. 2009. A fast update scheme for TCAM-based IPv6 routing lookup architecture. In *2009 15th Asia-Pacific Conference on Communications*. IEEE, 857–861.

[5] Md Iftakharul Islam and Javed I Khan. 2021. CP-TRIE: Cumulative popcount based trie for ipv6 routing table lookup in software and ASIC. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 1–8.

[6] Donghong Jiang, Yanbiao Li, Yuxuan Chen, Jing Hu, Yi Huang, and Gaogang Xie. 2023. Heuristic Binary Search: Adaptive and Fast IPv6 Route Lookup with Incremental Updates. In *Proceedings of the 7th Asia-Pacific Workshop on Networking*. 47–53.

[7] Fuliang Li, Jiahai Yang, Xingwei Wang, Tian Pan, Changqing An, and Jianping Wu. 2016. Characteristics analysis at prefix granularity: A case study in an IPv6

network. *Journal of Network and Computer Applications* 70 (2016), 156–170.

[8] Yuqiang Li, Fan Zhou, Xiaoxiang Zhu, and Jianming Liao. 2022. An IPv6 Routing Lookup Algorithm Based on Hash Table and HOT. In *2022 5th International Conference on Information Communication and Signal Processing (ICICSP)*. IEEE, 397–402.

[9] MAWI. [n. d.]. MAWI Working Group Traffic Archive. https://mawi.wide.ad.jp/mawi/. ([n. d.]).

[10] RIPE. [n. d.]. Route collectors. https://ris.ripe.net/docs/route-collectors/#bgp-timer-settings. ([n. d.]).

[11] V. Srinivasan and G. Varghese. 1999. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems* 17, 1 (Feb. 1999), 1–40. https://doi.org/10.1145/296502.296503

[12] Thibaut Stimpfling, JM Pierre Langlois, Normand Bélanger, and Yvon Savaria. 2018. A low-latency memory-efficient IPv6 lookup engine implemented on FPGA using high-level synthesis. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 402–411.

[13] David E. Taylor and Jonathan S. Turner. 2007. ClassBench: A Packet Classification Benchmark. *IEEE/ACM Transactions on Networking* 15, 3 (June 2007), 499–511. https://doi.org/10.1109/TNET.2007.893156

[14] David E Taylor, Jonathan S Turner, John W Lockwood, Todd S Sproull, and David B Parlour. 2003. Scalable IP lookup for Internet routers. *IEEE journal on selected areas in communications* 21, 4 (2003), 522–534.

[15] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. 1997. Scalable high speed IP routing lookups. In *Proceedings of the ACM SIGCOMM'97 Conference on Applications, technologies, architectures, and protocols for computer communication*. 25–36.

[16] Tong Yang, Gaogang Xie, YanBiao Li, Qiaobin Fu, Alex X Liu, Qi Li, and Laurent Mathy. 2014. Guarantee IP lookup performance with FIB explosion. In *Proceedings of the 2014 ACM Conference on SIGCOMM*. 39–50.

[17] Chunyang Zhang and Gaogang Xie. 2022. Using XorOffsetTrie for high-performance IPv6 lookup in the backbone network. *Computer Communications* 181 (2022), 438–445.