Efficient IPv6 Router Interface Discovery

Tao Yang and Zhiping Cai
College of Computer Science and Technology, National University of Defense Technology, China
E-mails: {*yangtao97, zpcai*}*@nudt.edu.cn*

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621168

***Abstract*—Efficient discovery of router interfaces on the IPv6 Internet is critical for network measurement and cybersecurity. However, existing solutions commonly suffer from inefficiencies due to a lack of initial probing targets (seeds), ultimately exhibiting limitations on large-scale IPv6 networks. Therefore, it is imperative to develop a methodology that enables the efficient collection of IPv6 router interfaces with limited resources, considering the impracticality of conducting a brute-force exploration across the extensive IPv6 address space. In this paper, we introduce Treestrace, an innovative asynchronous prober specifically designed for this purpose. Without prior knowledge of the networks, this tool incrementally adjusts search directions, automatically prioritizing the survey of IPv6 address spaces with a higher concentration of IPv6 router interfaces. Furthermore, we have developed a carefully crafted architecture optimized for probing performance, allowing the tool to probe at the highest theoretically possible rate without requiring excessive computational resources. Real-world tests show that Treestrace outperforms state-of-the-art works on both seed-based and seedless tasks, achieving at least a 5.57-fold efficiency improvement on large-scale IPv6 router interface discovery. With Treestrace, we discovered approximately 8 million IPv6 router interface addresses from a single vantage point within several hours.**

***Index Terms*—IPv6, Internet-wide Scanning, Network Measurement**

I. Introduction

The growing accessibility of IPv6-capable devices reflects the extensive deployment of IPv6 and signals the onset of an IPv6 era on the Internet. Since 2012, Google's users access the services via IPv6 has experienced a significant surge, skyrocketing from below 1% to reach a remarkable 40% [1]. A significant proportion of the top websites, currently estimated at 21.1%, have already adopted IPv6 at the time of writing this paper [2], [3].

The IPv6 routing infrastructure is a crucial component of today's Internet architecture, making the fast collection of its interface addresses greatly helpful for network measurement and cybersecurity, as 1) network topology mappings necessitate comprehensive coverage of router interfaces to depict interconnectivity among Internet nodes [4], [5] and 2) IPv6 target generation [6]–[8] relies on seed addresses from multiple sources (e.g., router interface addresses [9], [10]) for unbiased scanning in the IPv6 address space; furthermore, efficient discovery of IPv6 router interfaces enables network and security researchers to address various issues, including safeguarding critical infrastructure [11]–[13] and troubleshooting network connectivity problems [14], [15], thus minimizing network downtime and ensuring smooth system operations, while understanding the distribution of Internet router interfaces is crucial for applications such as anonymization techniques [16], enhancing content distribution [17], and network census [18].

IPv6, however, poses a challenge in this field due to its vast address space. In IPv4 networks, a comprehensive collection of router interfaces can be easily achieved by tracerouting all the IPv4 addresses, and numerous outstanding related works have made significant progress in the IPv4 field. Related solutions commonly employ variations of traceroute techniques to achieve higher probing speeds for IPv4 topology mappings, thereby exposing the numerous IPv4 router interface addresses. For example, Yarrp [19] randomly permutes an input $\texttt{IP} \times \texttt{TTL}$ space to map paths to all IPv4 "/24" networks in a stateless manner. Flashroute [20] builds upon the idea originated in Doubletree [21], exploring routes in a backward direction and ceasing probing when detecting routing convergence. However, the direct application of the aforementioned IPv4-oriented strategy, i.e., straightforwardly tracerouting the entire IPv6 Internet addresses in a brute-force manner, is considered impractical for conducting an IPv6 survey of router interfaces.

Indeed, some endeavors [11], [22] have been made in the discovery of IPv6 router interfaces for the purpose of achieving maximum coverage of the IPv6 routing infrastructure. Yarrpv6 [4] (a.k.a., IPv6-compatible Yarrp) employed a randomized algorithm for probing all hops along the paths to the given destinations, ultimately revealing over 1.3 million IPv6 router interface addresses by utilizing a set of high-quality seed addresses. However, this solution exhibits limitations in IPv6-wide tasks of router interface discovery when the initial IPv6 addresses are unavailable, as it is difficult to collect a sufficient number of unbiased seed addresses on the Internet-wide scale. This presents a challenge in achieving comprehensive coverage of the routing infrastructures across the IPv6 Internet. Therefore, it is imperative to develop an IPv6 router interface discovery method on a massive scale that eliminates the need for initial seeds, i.e., seedless.

To achieve this goal, we propose Treestrace, an asynchronous tool specifically designed for fast Internet-wide IPv6 router interface discovery. Without prior knowledge of the networks, Treestrace can simultaneously explore numerous IPv6 prefixes, such as the global-scale IPv6 BGP prefixes [23], to efficiently collect router interface addresses within restricted budgets, i.e., the number of allowed probing attempts. As a dynamic probing methodology, Treestrace considers the fundamental insight that expensive budgets for probing should prioritize areas of the IPv6 address space with higher rewards, given

the limited practicability of probing every address or "/64". Instead of allocating the entirety of the probing budgets to IPv6 prefixes without discrimination, Treestrace splits the complete process of router interface discovery into multiple phases at a predetermined interval of budget consumption. Within each probing round, Treestrace exercises prudent discretion in adjusting the allocation of budgets for each IPv6 prefix by taking into account the rewards derived from their previous probing results. In essence, prefixes that have demonstrated superior performance previously are allocated higher budgets, increasing the likelihood of discovering more IPv6 router interfaces in subsequent rounds. However, implementing the above dynamic probing strategy is nontrivial on real-world IPv6 networks.

Implementing sequential probing of all hops along a path to each destination for all IPv6 prefixes, like traditional traceroute tools, is infeasible [19]. The influx of probe packets targeting a single IPv6 prefix within a short time would trigger packet rate-limiting and overload the networks, leading to inconsistent measurement results. Therefore, Treestrace introduces an innovative weighted random sampling method, which can randomly permute the transmission order of probe packets while honoring the respective budgets allocated to the associated IPv6 prefixes, without imposing excessive computational or storage demands. To the external Internet, the probing traffic appears to be widely distributed across all the IPv6 prefixes, thereby avoiding overloading routers or links. More details will be explained in § III-B.

The contribution of the paper could be summarized as follows:

- An innovative asynchronous prober, Treestrace, has been developed for efficient discovery of IPv6 router interfaces. As its implementation requires neither IPv6 seeds nor excessive computation and storage, Treestrace can quickly collect IPv6 router interfaces on an Internet-wide scale with a newly proposed dynamic probing strategy.
- A novel weighted random sampling algorithm can effectively promote the solution of allocating limited resources among competing choices to maximize expected gain. This algorithm is particularly well-suited for large-scale Internet measurements, where efficient resource allocation is crucial.
- An address analysis of Internet-wide IPv6 router interfaces is fulfilled by Treestrace. Real-world tests show that Treestrace significantly improves both the discovery rate and efficiency of IPv6 router interface discovery compared to state-of-the-art works, enabling us to discover around 8 million IPv6 router interfaces from a single vantage point within a few hours.

II. Preliminaries

A. Background

IPv6 addressing system and routing prefix. The addressing system employed in IPv6 differs from its predecessor, IPv4, and is characterized by the adoption of a 128-bit address space [24], [25], allowing for the allocation of at least one IPv6 global unicast address to Internet-connected devices. However, the addressing of packet forwarding in IPv6 does not utilize all 128 bits of the address space [26]. Instead, the IPv6 addressing architecture follows a hierarchical structure [25], comprising a routing prefix, a subnet identifier, and an interface identifier, as depicted in Fig. 1. As the basic implementation in IPv6, primitive routers adhere to the "longest-match-first" rule when forwarding IPv6 packets based on the routing prefix of the destination [27]. As a result, the IPv6 addressing boundary is fixed at a length of 64 bits [28], with the "/64" prefixes serving as the basic unit for assigning address space size as specified by RIPE NCC [29].

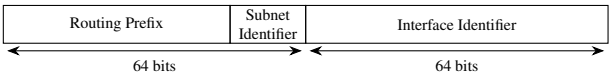


Fig. 1: IPv6 128-bits address, where routing prefix size varies.

IPv6 address pattern. The lower 64 bits of an IPv6 global unicast address are assigned to the interface identifier (IID) to identify the specific device within a network [30]–[32]. Accordingly, typical IIDs of IPv6 addresses can be categorized into patterns as illustrated in Tab. I, where other patterns (e.g., embed-Port or embed-wordy) are quite rare in our results.

TABLE I: Typical IPv6 Address IID Patterns

Patterns	IID Examples	Comments
EUI64[a]	0250:56ff:fe89:49be	embed MAC address 00:50:56:89:49:be and then flip 7_{th} bit
Embed-IPv4	0012:0122:0126:0072	embed IPv4 address 12.122.126.72
Low-byte	0000:0000:0000:f1b7	all zeros except the lower bytes
Pattern-bytes	0021:2222:0001:0001	more than two bytes of zeros
Randomized[b]	10de:51e8:eb66:7583	pseudorandom

[a]: The first three bytes represent Organizationally Unique Identifier [30].
[b]: Privacy Extensions for Stateless Address Autoconfiguration [31].

ICMP rate-limiting. This is a necessary function for IPv6 nodes to control the generation of ICMP error messages, and its widespread implementation effectively reduces the risk of network flooding with ICMP packets [33]. However, aggressive ICMP rate-limiting in IPv6 presents challenges for traceroute-based measurements, e.g., router interface discovery [4], [11], [14]. Therefore, a well-designed approach is needed to manage probing traffic and mitigate inconsistent measurement results caused by ICMP rate-limiting.

Interaction-based measurement. It, also known as the reinforcement learning-based network measurement, has been widely employed in network scanning methods such as 6Hit [6] and 6Scan [7], [34]. 6Hit pioneeringly utilized the concept of reinforcement learning for IPv6 target generation, resulting in significant improvements in active IPv6 address discovery. Similarly, 6Scan incorporated the reinforcement learning algorithm into an asynchronous scanning tool, greatly enhancing scanning efficiency. In summary, interaction-based network measurement prioritizes the exploration of those high-reward targets, enabling optimizing cost-effectiveness through cautious resource allocation and redundancy elimination.

B. Related Works

Currently, two production measurement platforms, CAIDA's Ark [35] and RIPE Atlas [36], actively engage in continuous IPv6 topology mapping. These systems utilize traditional tools (i.e., traceroute6 and scamper [37]) for sending probes directed towards the `::1` address in each IPv6 prefix listed in the global BGP table (with CAIDA Ark additionally probing a random address within each prefix). From CAIDA Ark databases which performed the measurements from multiple vantage points in July 2023, approximately 400K unique IPv6 router interfaces were extracted. In 2015, Rohrer et al. [5] uniformly sampled the IPv6 Internet by tracerouting to an address within every "/48" in all the announced "/32s", in order to characterize the distribution of IPv6 interface addresses in the wild, uncovering 128K router interfaces with 400M probes (traces). Inspired by Zmap [38], Yarrpv6 [4] introduced a stateless probing technique, significantly enhancing the parallelism of its random probing strategy, thereby collecting over 1.3M IPv6 router interface addresses. However, this design necessitates an increase in the volume of required probes, because the absence of state hampers the prober's ability to effectively utilize response feedback for adjusting its probing direction. Consequently, Yarrpv6 exhaustively sends probes to every possible hop for each destination, disregarding the uneven distribution of interface addresses. On a side note, FlashRoute [20], a Doubletree-based asynchronous traceroute tool exclusively oriented towards IPv4 topology, has been enhanced by its developers to explore individual IPv6 addresses [39], although there is not yet literature reporting this. However, the Doubletree strategy [21] for removing redundancy is not highly practical for discovering IPv6 router interfaces without the IPv6 hitlists, which will be discussed in § V.

C. Router Interface Discovery Model

In order to avoid potential misconceptions arising from a stereotypical understanding of IPv4 networks, the model of router interface discovery utilized by Treestrace can be presented as following: Considerring the 64-bit addressing boundary [28] (see § II-A), Treestrace would not typically expect that traceroutes to multiple addresses within the same "/64" will yield different topologies or router interfaces, as indicated by related studies [4]. Therefore, a probe in Treestrace can be uniquely defined by a combination of a "/64" and an initial Hop Limit value, namely probed pair $\langle /64, \texttt{Hop} \rangle$. By selectively traversing such a $/64 \times \texttt{Hop}$ space within the budget limitation, Treestrace can efficiently produce the probe packets located into the expected IPv6 areas, which are explained in detail in § III.

Upon the selection of a probed pair, the ensuing matter involves the creation of a 64-bit IID to accompany the provided "/64" for the purpose of establishing a comprehensive IPv6 address, designated as the probing destination. Treestrace employs a series of predetermined rules to generate IIDs, with the objective of augmenting the number of available bits for probe encoding, thereby expanding the measurement capacity of Treestrace, as delineated in § III-A. Additionally, majority of destination addresses in Treestrace's probes remain inactive, thereby enabling us to concentrate solely on the ICMPv6 error responses (Destination Unreachable or Time Exceeded) received from router interfaces. Therefore, this simplifies the probing logic of Treestrace.

Some related works, such as CAIDA Ark [35] or Yarrpv6 [4], commonly employed IPv6 addresses with "lowbyte1" IIDs (`::1` with zero compression) as the probing targets for this traceroute-like measurement. However, we intentionally exclude them to ensure that Treestrace's Internet-wide measurements remain non-intrusive, as these "lowbyte1" addresses are typically assigned to devices hosting critical user services.

Based on the aforementioned model, we can efficiently perform the discovery of IPv6 router interfaces, by probing the generating IPv6 addresses with corresponding Hop Limit values, provided that the probe states are effectively maintained and the probing traffic is well managed. We are thus committed to accomplishing these goals in this work.

III. Treestrace Design

Treestrace tool is optimized to undertake the simultaneous exploration of a broad expanse of IPv6 prefixes, aiming to maximize the collection of IPv6 router interface addresses within limited probing budgets. It is written in Go, exclusively utilizing the native standard library, and is compatible with various UNIX-like platforms.

As such capability has not yet been achieved in the realm of IPv6 router interface discovery so far, the main innovation of Treestrace can be summarized as follows: *While preserving the advantage of high parallelism through the decoupling of sending and receiving tasks, Treestrace also maintains the capability to adjust the probing direction by leveraging feedback from prior responses.*

Fig. 2 presents the system architecture of Treestrace. The receiving task can be swiftly implemented by performing duplicate elimination (using Bloom filters) and asynchronously updating reward records for the corresponding prefixes (using atomic operations) upon capturing the response packets. However, the implementation of the sending thread poses a significant challenge. It must effectively manage probe states for a large set of IPv6 prefixes and ensure the rapid transmission of probing packets to fully utilize the available bandwidth. To address this challenge, Treestrace introduces a series of novel techniques. The following sections will outline the implementation details for each of these mechanisms.

A. Probe Encoding

Similar to previous asynchronous tools (e.g., Yarrp [19] and Zmap [38]), we encode all the necessary measurement information into the probe packet header, which is then returned in the response packets. In the case of ICMPv6 echo request packets, it is worth highlighting that the fields of ICMPv6 Identifier and Sequence Number can be modified to carry additional information, providing a potential capacity of up

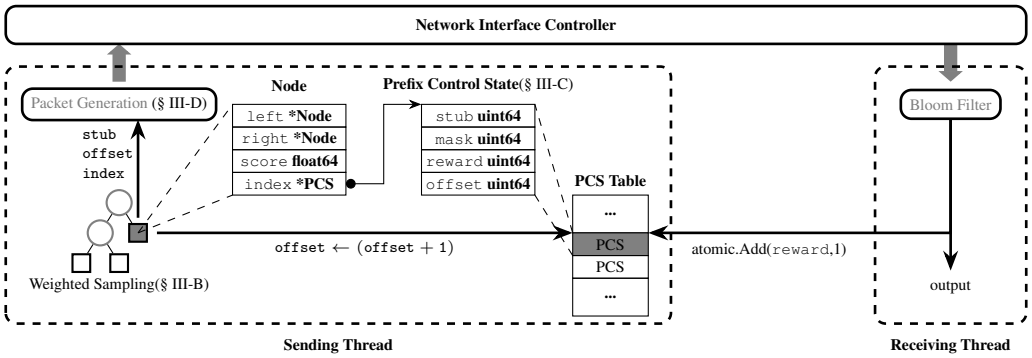


Fig. 2: The architecture of Treestrace.

to 32 bits for probe encoding purposes, as depicted in Fig. 3. Specifically, we allocate the lower 8 bits to represent the initial Hop Limit of the probe, while the remaining 24 bits are used to indicate the index of the originating prefix of the probe.

Furthermore, Treestrace incorporates the interface identifiers (IIDs) of destination addresses to facilitate supplementary probe encoding, thus enabling additional measurement functions. Specifically, the 64-bit IID of a destination address could be divided into three distinct components: a 32-bit timestamp is utilized for round-trip-time (RTT) computation, a 16-bit instance identifier ensures that received packets are authentic responses to Treestrace's probes, and a 16-bit checksum is calculated over the upper 112 bits of IPv6 targets, thereby enabling the detection of any modifications to the IPv6 destination address, e.g., due to the middleboxes. Actully, our results have identified a negligible quantity of responses (not zero) with modified addresses.

In real-world networks, it is not uncommon for routers to deviate from the requirements outlined in RFC4443 [33] and discard the probe payload when generating ICMP Unreachable/Time-exceeded error messages. This behavior can present challenges in attributing the discovered IPv6 router interfaces to corresponding IPv6 prefixes if the probing information (e.g., Prefix Index or Hop Limit) were encoded in the volatile payload of probing packets. To overcome this issue, Treestrace has been specifically designed to integrate all the essential information for router interface discovery into the headers of probe packets, as discussed above. Additionally, the reserved payload of the probe packet is exclusively utilized to include the website for declaring the purpose of our measurements (see Fig. 3). This approach ensures that the probing information can be extracted from the responses received from IPv6 router interfaces, thereby eliminating the need for extensive storage and excessive computation required to maintain this probe information locally.

B. Dynamic Probing Strategy

The development of the Treestrace tool is driven by the fundamental facts outlined below: it is nearly impossible to brute-force explore every probed pairs $\langle /64, \texttt{Hop} \rangle$ (see § II-C) to achieve a comprehensive collection of IPv6 router interfaces on the real-world networks. Therefore, a viable methodology for approximating the ground truth is to pursue the maximum

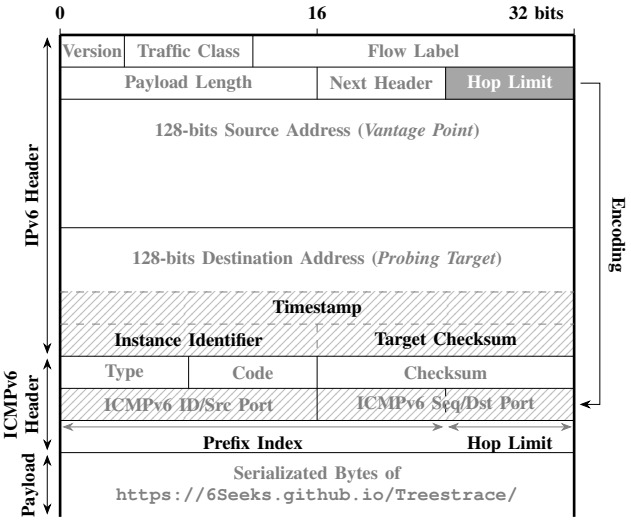


Fig. 3: Treestrace encodes information in the (line-filled) fields of probing ICMPv6 packets.

number of router interface addresses with a limited probing budget.

Assuming, without loss of generality, that the input for Treestrace consists of n multisize IPv6 prefixes whose address volumes (number of unique addresses) and probing budgets are respectively $\{v_1, \ldots, v_n\}$ and $\{x_1, \ldots, x_n\}$. Formally, the problem can be restated as a combinatorial optimization problem [40], whose model $P = (X, f, g)$ comprises of:

- A finite set of candidate solutions X, that are defined over a series of vectors of decision variables $x \in X$. Specifically, the i_{th} prefix of the solution x is allocated with a total of x_i probing budgets for a particular solution, represented by $x = \{x_1, \ldots, x_n\}$.
- The object function f, that produces the number of discovered IPv6 router interfaces $f(x)$, given a solution vector $x \in X$.
- The constraint function g, that presents the limitation of probing budget b in this task, i.e., $g(x) : x_i + \cdots + x_n \leq b$.

Accordingly, the goal of IPv6 router interface discovery can be formulated as follows: maximize $f(x)$, subject to $g(x)$. And, it can be concluded that a solution $x^* \in X$ is the global optimum if and only if the following condition holds: $\forall x \in$

$X, f(x^*) \geq f(x), g(x) < b$.

Obviously, there are various allocation strategies for the probes sent to different prefixes. It is important to note that the random strategy-based prober, such as Yarrpv6 [4], [19], generates a solution denoted as $x^s = x_1^s, \ldots, x_n^s$ where $x_i^s \propto v_i$. However, this solution is unlikely to be the global optimum due to the uneven distribution of router interface addresses in the IPv6 address space. A more optimal approach would involve allocating the budget to prefixes based on their respective global rewards. However, the lack of prior knowledge about the distribution of routing devices in the IPv6 network makes it impossible to determine the actual global rewards of the input prefixes.

To resolve it, Treestrace utilizes an interaction-based probing strategy, which leverages feedback from previous probes to dynamically adjust the allocation of the probing budget across all prefixes. In this approach, prefixes with high rewards are prioritized for subsequent probes, while prefixes with low rewards are probed more cautiously. While the concept is straightforward, its implementation presents a significant challenge as it necessitates meticulous adjustment of budget allocation and effective management of the probing traffic, all while maintaining a high probing rate.

Algorithm 1 Weighted Sampling based on Huffman Coding

Require: Root of coding tree R
Ensure: Leaf node of coding tree L incorporating the exact prefix for probe generation

```
C ← R                                  # pointer of current node
I ← Rand.Uint64()                      # initialization of randam bits
repeat
    if I ⊗ 1 = 0                       # bitwide xor for last bit then
        C ← C.left
    else
        C ← C.right
    end if
    I ← I × 2^-1                       # bitwise right shift
until C.isLeaf()
L ← C
```

In Treestrace, a complete binary tree is utilized for the solution, as illustrated in Fig. 4, where each leaf node represents a distinct prefix. This structure is inspired by the well-known Huffman coding tree, which is designed to transform assigned probabilities associated with the leaf nodes into varying coding lengths, as reflected by their depth in the binary tree. Conversely, we can also derive the probability from the leaf nodes in the tree, enabling weighted random sampling. Specifically, we generate a random bit stream and sequentially follow its bit values to traverse the binary tree until we reach a designated leaf node, as illustrated in Algorithm 1. This leaf node contains the exact prefix used to generate a probe packet during this sampling process. In this scenario, the random walk starting from the root node is more inclined to encounter high-reward prefixes, which are located on the shallow nodes (e.g., Prefix No. 1 with a score of 7.67), rather than the low-reward ones hidden deep within the binary tree (e.g., Prefix No. 7 with a score of 0.18), as depicted in Fig. 4.

After proposing an efficient weighted random sampling approach, the remaining issue pertains to the rapid *re-initialisation* of the corresponding binary tree according to the new weights, each time we consumed k probes (empirically $K = 100000$). Estimating the exploration value of a prefix by directly using the ratio between the reward (i.e., the number of router interface addresses ρ) and the cost (i.e., the number of probes σ) is inappropriate due to the order-of-magnitude difference in the address volumes between prefixes of varying sizes. Hence, the prefix score is denoted as $\frac{\rho}{\log_2 \sigma}$ for the corresponding nodes, which are then initialized into a priority queue (i.e., a minimum heap) as shown in Algorithm 2. Iteratively, the priority queue pops two nodes with the minimal scores, merges them into one, and reinserts the merged node back into the queue until only one node remains, i.e., the root node of the coding tree. According to the aforementioned construction process of the coding tree, the sampling probability can be approximately expressed as $2^{-\lceil \log_2(1/f) \rceil}$ and at least as $2^{-2\lceil \log_2(1/f) \rceil}$ for a given prefix with a score percentage of f [41]. Moreover, the sampling process has a complexity that is bounded by a constant (specifically around 64, due to the 64-bit addresseing boundary in IPv6), denoted as $O(1)$. Additionally, the time and space complexities of constructing the coding tree are both $O(n \log n)$, making it easily implemented within several milliseconds given the capabilities of modern computers.

Algorithm 2 Huffman Coding Tree Initialisation

Require: A set of node $\{\mathtt{N}_1, \mathtt{N}_2, \ldots, \mathtt{N}_n\}$
Ensure: Root node of coding tree R

```
H ← Heap({N_1, N_2, ..., N_n})     # min heap priority queue
while H.size() > 1 do
    N ← Node()                     # initial a new (internal) node
    N.left = N_left ← H.pop()      # pointers of child nodes
    N.right = N_right ← H.pop()
    N.score ← N_left.score + N_right.score
    H.push(N)
end while
R ← H.pop()
```

Unlike existing interaction-based approaches that distribute budgets to prefixes based solely on the Boltzmann distribution derived from their rewards [6], [34], Treestrace proposes a weighted random sampling technique that offers enhanced management of probing traffic. In traceroute-like measurements for router interface discovery, it is crucial to evenly distribute the probing traffic since exploring the route of each target "/64" within a specific prefix necessitates *dozens of* probe packets. Nonetheless, after allocating the budget, the existing interaction-based solutions send probes in a sequential manner, one prefix at a time, which might lead to inconsistent measurement results due to the influx of probing traffic triggering the packet rate-limiting of the target networks. To tackle this issue, existing approaches have no choice but to employ a buffer for the transient preservation of probe

#	Prefix	Score	Bit Stream	Selection Probability
1	2a02:60::/28	7.67	*******0	2^{-1}
2	2001:278::/32	2.29	******11	2^{-2}
3	2408:2000::/24	0.77	****0001	2^{-4}
4	2408:805c::/30	0.73	****1001	2^{-4}
5	2001:5000::/21	0.70	****0101	2^{-4}
6	2804:b:c002::/47	0.32	***01101	2^{-5}
7	2c0f:fd98:8110::/48	0.18	***11101	2^{-5}

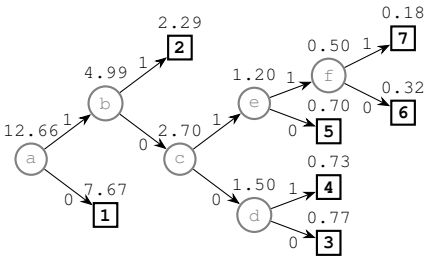


Fig. 4: A Huffman coding tree for weighted random sampling on 7 various prefixes. Note that we use the wildcard symbol "*" to denote the varying bits.

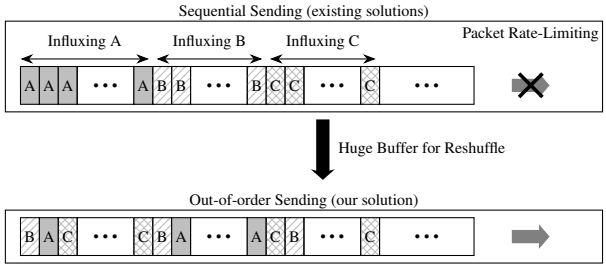


Fig. 5: The comparison of sending orders between existing methods and ours, involving three IPv6 prefixes A, B and C.

packets for the purpose of reshuffling, as described in Fig. 5, ultimately resulting in a substantial demand for storage and time. In contrast, the dynamic probing strategy of Treestrace inherently supports the transmission of probe packets among the prefixes in an out-of-order manner, thereby preventing network overload.

C. Control State

Treestrace requires control state to support its probing logic.

Static data structures. For each input prefix, it maintains a group of states to keep track of the probing progress and record the feedbacks of probes. Specifically, Treestrace utilizes two `uint64` integers, namely `stub` and `mask`, to define the range of "/64s" for a specific input prefix. For example, the given input prefix `2001:1234::/32` can be represented by a `stub` (`0x2001123400000000`) and a `mask` (`0xffffffff`), thereby establishing a one-to-one correspondence between each "/64" and a unique integer value ranging from 0 to `mask`. By assigning these aforementioned states to each input prefix, we can easily monitor the entire probing progress and perform subsequent discovery of router interface addresses. Two other states of the input prefixes, namely `reward` and `offset`, are utilized to document the number of captured router interface addresses and the number of consumed probes, respectively, in order to estimate the prefix scores as introduced in § III-B. In the implementation of Treestrace, to facilitate retrieval and organization of the various prefix states, we consolidate all the aforementioned states into a single structure, namely "Prefix Control State" (PCS) block, as illustrated in Fig. 2. Following this, we instantiate all input prefixes as a collection of PCS blocks, which collectively compose a static PCS table.

Dynamic data structures. While Treestrace's dynamic probing strategy, based on weighted sampling, can be implemented through a random walk on the established Huffman coding tree as illustrated in § III-B, efficiently retrieving the corresponding PCS block when reaching a leaf node remains a significant challenge. The straightforward approach of copying the states of the PCS block into the associated 2-ary node may result in out-of-sync data if the sending and receiving threads simultaneously modify the states of the same prefix. To address this issue, Treestrace introduces a state of PCS `index` into the node structure, decoupling the (dynamic) coding tree nodes from the (static) PCS blocks. This ensures synchronization of control states and eliminates the overhead of frequent copies. As a result, Treestrace can swiftly retrieve the PCS block corresponding to the associated prefix upon sampling a leaf node, enabling the rapid generation of a probe packet with the PCS `index` populated into the 24-bit Prefix Index fields of its header.

Overall, the complete structure, consisting of the PCS table and coding tree, exhibits remarkably low RAM occupancy, such as approximately 700 MBytes when concurrently conducting the Internet-wide measurements on one million prefixes while the Route Views comprehensive databases consist of only 49130 IPv6 BGP prefixes after removing the overlaps [23]. Therefore, this amount easily falls within the capacity of modern computers.

D. Randomized Probe Generation

Treestrace introduces the technique of randomized probe generation to address the following issue: When a high-reward prefix is assigned to a *too* shallow leaf node of the coding tree (e.g., due to incomplete exploration convergence or significant variations between input prefixes), there is a tendency to repeatedly select the same prefix for generating probes within a short period. Therefore, it is crucial to evenly distribute the probing traffic for each individual prefix to mitigate packet rate limitations.

Unlike the traditional traceroute technique that sequentially probes all hops along a path to a target, Treestrace traverses the space of $/64 \times \texttt{Hop}$ for a given prefix in a random permutation. Specifically, Treestrace employs the FNV-1 hash function to encrypt the `offset` state (the number of probes consumed by this prefix), with each ciphertext corresponding to a unique probed pair $\langle /64, \texttt{Hop} \rangle$, as illustrated in Fig. 6. Due to the avalanche effect in hash function, i.e., the ciphertext changes significantly even if the input is changed slightly, Treestrace could send probes to a "/64" A with a Hop Limit of 12, then to B with a Hop Limit of 7, subsequently to C with a Hop Limit of 23, and so on until the probing budget is exhausted. In this way, the probing traffic is spread across the entire prefix rather than a narrow range, to the external Internet.

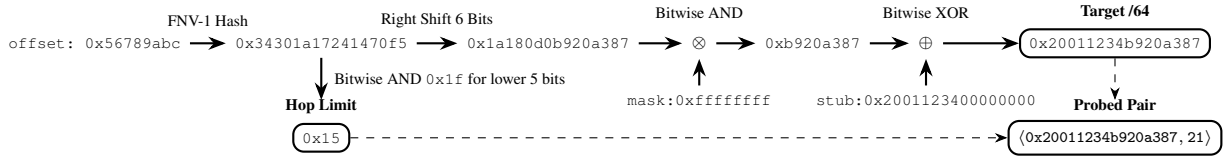


Fig. 6: The probe generation process of Treestrace about prefix `2001:1234::/32`. Treestrace excludes probed pairs with Hop Limit values of ≤ 1 to safeguard the local routers.

TABLE II: Active Internet-wide Topology Discovery Results With 100-Kpps Uplinks

Approach	Input	Router Interfaces	Probing Packets	Involved ASes	Discovery Rate (‰)	Time Cost (Second)	Probing Speed	Efficiency (Num. per Sec.)
Treestrace	Hitlist Seeds	578415	1.00×10^7	4825	**57.84**	331.25	30.19Kpps	**1746.16**
Flashroute		293598	2.57×10^7	7369	11.42	1549.2	16.59Kpps	189.52
Yarrpv6		326010	8.03×10^7	7786	4.06	7743.6	10.37Kpps	42.10
Treestrace		2354716	1.00×10^8	6531	**23.55**	2313.7	43.22Kpps	**1017.73**
Treestrace	BGP Prefixes	642551	1.00×10^8	8719	**6.43**	1921.8	52.03Kpps	**334.35**
Yarrpv6		180331	1.29×10^8	8979	1.39	11840.5	10.92Kpps	15.23
Treestrace		7703499	1.00×10^9	9066	**7.70**	19549.8	51.15Kpps	**394.04**

IV. Evaluation

We evaluate the performance of Treestrace and existing solutions (i.e., Yarrpv6 [4] and Flashroute [39]) on real-world networks.

A. Experiment Setups

1) Data Inputs: To establish the fair and unbiased performance comparisons of real-world tests on a large scale, incorporating two distinct IPv6 datasets as inputs is essential:

- IPv6 hitlist seeds. Gasser et al. [10], [42] provided the IPv6 hitlist [9], which comprised about 7M active IPv6 addresses (a.k.a., seeds), obtained through active scanning and passive collection, involving 17891 public ASes.
- IPv6 BGP prefixes. The Route Views project [23] offers up-to-date BGP information on the worldwide routing system, observed from backbones and locations across the Internet. Its database contains 49130 orthogonal IPv6 BGP prefixes of various sizes, covering 30789 public ASes.

Note that existing methods cannot explore multiple prefixes concurrently, as they solely accept the individual addresses or single IPv6 prefixes as valid inputs. Therefore, 40 million target addresses are downsampled from the above BGP prefixes as input for Yarrp, following related work [4].

2) Probing Scale: Controlling the probing scale of existing methods presents a formidable challenge due to the absence of support for configuring probe limitations, unless modifications are made to the number of input seeds or prefixes. Abruptly terminating the process of router interface collection upon reaching the budget limitation is not feasible, as it may compromise the integrity of the measurement results obtained by existing methods. For a fair comparison, we will not impose any restrictions on probing packets until the baselines finish the probing processes.

To avoid misconceptions, we assigned two distinct budgets to each experiment for the comprehensive performance evaluation of Treestrace, with the intervals for its dynamic probing adjustment empirically set to 100K probes per round.

3) Probing Protocol: Any IPv6 packet can be utilized for discovery of IPv6 router interface addresses, as only ICMPv6 error messages are considered valid responses from router interfaces. Previous studies have demonstrated that probing with ICMPv6 results in approximately 2.2% and 2.1% more discovered interface addresses compared to using UDP and TCP, respectively [4]. In our comparative experiments, we utilized the ICMPv6 echo request as the probing payload because it is designed for diagnostic purposes and is less intrusive than UDP and TCP probes, as supported by previous studies [11].

4) Evaluation Metric: Following the prior works [7], to quantitatively evaluate the ability of existing methods, we propose two metrics, namely discovery rate and efficiency. Formally, a methodology can be proposed to discover the set of router interface addresses, namely τ, within a limited budget b (i.e., the number of probing packets). The discovery rate of IPv6 router interface discovery can be expressed as $\frac{|\tau|}{b}$, where $\tau \subset A$. Similarly, given the time consumption t required for the aforementioned task, the efficiency in IPv6 router interface discovery can be expressed as $\frac{|\tau|}{t}$, where $\tau \subset A$.

B. Real-world Performance

In May 2023, we conducted large-scale router interface discovery experiments of Treestrace, employing 100K probes per phase (see § I), for performance comparison with the baselines. However, Flashroute is characterized by its redundancy elimination, using only active addresses as valid inputs. This renders it incompatible with measuring IPv6 BGP prefixes, thus excluding Flashroute from the performance comparison.

1) Discovery Rate: In the (seed-based) experiments using IPv6 hitlist seeds, all the methods effectively discover the IPv6 router interface addresses, and the redundancy elimination of Flashroute actually reduces the probe consumption compared to the randomized Yarrpv6. However, compared to state-of-the-art methods, Treestrace not only achieves the highest discovery rates of $23.55‰ \sim 57.84‰$ with various probe budgets but also discovers more IPv6 router interfaces

with fewer measurement resources (e.g., Treestrace uncovered over 578K router interfaces with only 10M probes while Flashroute uncovered 293K's using 25.7M probes). Specifically, Treestrace achieves an average 3.56-fold improvement on discovery rate performance over existing solutions when the seeds are available.

In the seedless experiments using BGP prefixes, the vast search space and sparse distribution of active addresses present challenges for existing methods (e.g., Flashroute). However, even without active addresses as prior knowledge of the networks, Treestrace achieves discovery rates of $6.43‰ \sim 7.70‰$, which is an average 5.08-fold improvement compared to the baseline method. Additionally, in the billion-scale experiment, Treestrace reveals 7.7 million IPv6 router interfaces within a few hours, significantly enhancing our capability for IPv6 network measurements.

2) Efficiency: To evaluate the efficiency of Treestrace and the baselines in real-world discovery of IPv6 router interfaces, we also introduce the concept of "Probing Speed", i.e., the average probing rate throughout the entire probing procedure, calculated as the ratio of the total number of probing packets to the total time.

The results show that all tools performed well in terms of probing speed, but Treestrace consistently outperformed the others in all experiments. Specifically, Treestrace can discover over 1000 router interfaces per second when seeds are available, and it also maintains the high efficiency in seedless tasks of IPv6 router interface discovery. This can be attributed to the following reasons: 1) The dynamic probing strategy of Treestrace automatically adjusts the direction of router interface discovery to explore the areas of IPv6 address spaces with higher rewards, resulting in the discovery of more router interfaces. 2) The carefully designed optimizations in Treestrace reduce system complexity and improve the transmission of probing packets, enabling high probing speeds even on an entry-level VPS host. In comparison to existing works, Treestrace eventually achieved the $5.37\times \sim 9.21\times$ efficiency improvement on seed-based task and the $21.95\times \sim 25.87\times$ efficiency improvement for seedless task. This demonstrates the utility of Treestrace for large-scale IPv6 router interface discovery.

C. Address Analysis

In the address analysis, we integrated the results obtained from Treestrace in each scenario, namely $S_{\text{Treestrace}}$ for experiments on IPv6 hitlist seeds and $P_{\text{Treestrace}}$ for experiments on IPv6 BGP prefixes. Similarly, the baselines' results could be represented as $S_{\text{Flashroute}}$, S_{Yarrpv6}, and P_{Yarrpv6}.

1) Major Autonomous Systems: Tab. III presents the top 10 Autonomous Systems (ASes) along with the respective count of newly-discovered router interfaces from each result source mentioned above. It is evident that a small number of ASes dominated the majority of the router interfaces in the interaction-based methods, with $S_{\text{Treestrace}}$ standing out as the leading AS, contributing to approximately 70% of the discoveries. The concentration of discovered IPv6 router interface addresses is mainly attributed to Treestrace's autonomous adjustment mechanism of the search direction, resulting in more probing budgets assigned to those autonomous systems with complex topological structures including numerous interface addresses of routing infrastructures. Nevertheless, Treestrace prioritizes the exploitation of high-reward prefixes while not neglecting the exploration of other promising prefixes, as evidenced by the number of router interfaces discovered in each Autonomous System. For example, Treestrace can discover 5187 and 14319 interface addresses in the 10th autonomous system, respectively, which exceeds other baseline methods in both seed-based and seedless tasks. The findings indicate that Treestrace effectively strikes a balance between the depth and breadth of exploration while uncovering the distribution of IPv6 router interfaces.

To further demonstrate it, we visualized the router interfaces discovered by Treestrace within 197K BGP prefixes from RouteViews using the zesplot tool [43], where each pixel represents a BGP prefix. As depicted in Fig. 7, the results obtained by Treestrace cover a wide range of BGP prefixes. Additionally, $P_{\text{Treestrace}}$ involves more BGP prefixes than $S_{\text{Treestrace}}$ since it is free from the biases of initial seeds.

2) Overlaps: The overlaps of the router interfaces from different sources are a critical metric to evaluate the contribution of the methods to the new discoveries of the Internet-wide IPv6 routers. For this joint overlap analysis of router interfaces, we compare the results with the production measurement platforms that continuously perform active IPv6 topology mapping, such as CAIDA Ark [35]. Fig. 8 shows that Treestrace has contributed the most number of newly-discovered router interfaces in both seed-based and seedless tasks of router interface discovery, as these results have only 2% overlaps with other methods or platforms. Furthermore, the discoveries of existing solutions present a significant correlation with others. For example, 31% of router interfaces in CAIDA Ark and 64% of router interfaces in P_{Yarrpv6} have been reported in $P_{\text{Treestrace}}$. In summary, our solution exhibits an endeavor of comprehensive survey into IPv6 router interfaces, compared to existing works.

3) Address IID Patterns: As mentioned in § II-A, the Interface Identifiers of an IPv6 address can usually be customized by users either manually or automatically [30], [32], [44]. These interface identifiers are crucial for revealing important information about IPv6 network assets. Therefore, the results of address pattern classification on the discovered IPv6 router interface addresses are presented in Tab. IV. It is evident at once that the predominant portion of IPv6 router interfaces discovered across all sources comprises the Low-byte addresses, whereas the Byte-pattern and Randomized addresses constitute a minority percentage. Surprisingly, the addresses obtained through autoconfiguration, namely EUI64, Embed-IPv6, and Low-byte, jointly contribute to the corresponding ratio (approximately 80%) of router interfaces in all sources, thereby confirming the veracity of findings from the Treestrace and baselines.

-	Number	Ratio
AS3320	38600	13.15%
▲	24740	8.43%
AS4134	16721	5.70%
AS20940	9127	3.11%
AS16276	8684	2.96%
AS4812	8587	2.92%
AS14340	8094	2.76%
AS7922	6051	2.06%
AS26101	5307	1.81%
AS714	4179	1.42%
AS36647	4109	1.40%

(a) $S_{Flashroute}$

-	Number	Ratio
AS4812	32880	10.09%
AS4134	31207	9.57%
▲	26503	8.13%
AS134774	14244	4.37%
AS20940	10783	3.31%
AS3320	10225	3.14%
AS16276	9437	2.89%
AS14340	8103	2.49%
AS7922	6634	2.03%
AS26101	5281	1.62%
AS714	4324	1.33%

(b) $S_{Yarrpv6}$

-	Number	Ratio
▲	26664	14.79%
AS13335	11310	6.27%
AS4134	9800	5.43%
AS1136	4612	2.56%
AS8151	3796	2.11%
AS24560	3135	1.74%
AS174	2976	1.65%
AS45609	2912	1.61%
AS16509	2642	1.47%
AS7922	2543	1.41%
AS4837	2344	1.30%

(c) $P_{Yarrpv6}$

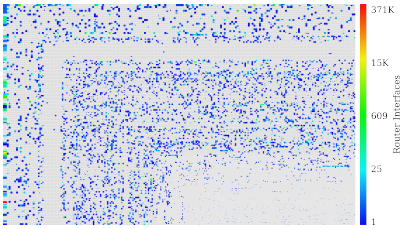
-	Number	Ratio
AS1136	1769690	74.34%
AS8422	319672	13.43%
AS2119	141164	5.93%
AS5607	14458	0.61%
▲	12192	0.51%
AS9136	9885	0.42%
AS4134	7015	0.29%
AS13037	6469	0.27%
AS203953	6194	0.26%
AS7018	5807	0.24%
AS2516	5187	0.22%

(d) $S_{Treestrace}$

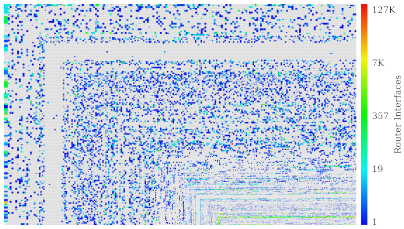
-	Number	Ratio
AS45609	3689317	47.23%
▲	1387384	17.76%
AS24560	1092186	13.98%
AS8151	588444	7.53%
AS38266	542015	6.94%
AS8422	127104	1.63%
AS17072	80117	1.03%
AS29555	57426	0.74%
AS28006	35291	0.45%
AS138754	25341	0.32%
AS55836	14319	0.18%

(e) $P_{Treestrace}$

TABLE III: Top 10 ASes of router interfaces discovered from each source (▲:unreported BGP Prefixes)



(a) $S_{Treestrace}$



(b) $P_{Treestrace}$

Fig. 7: All 197K BGP prefixes in RouteViews database, colored based on the number of IPv6 router interfaces, with *zesplot* tool.

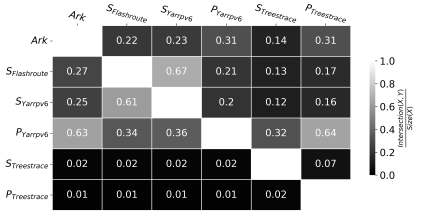


Fig. 8: Overlaps between IPv6 router interfaces from all the sources for each row.

TABLE IV: Address IID Patterns of Router Interfaces Discovered from Each Source

Source	Total Number	Address IID Pattern				
		EUI64	Embed-IPv4	Low-byte	Byte-pattern	Randomized
$S_{Flashroute}$	293598	42054 (14.32%)	33326 (11.35%)	174108 (59.30%)	14457 (4.92%)	21354 (7.27%)
$S_{Yarrpv6}$	326010	13170 (4.04%)	34617 (10.62%)	187375 (57.48%)	15812 (4.85%)	66468 (20.39%)
$P_{Yarrpv6}$	180331	10358 (5.74%)	38141 (21.15%)	101977 (56.55%)	10869 (6.03%)	10869 (6.03%)
$S_{Treestrace}$	2379657	804547 (33.81%)	24631 (1.04%)	1107524 (46.54%)	5271 (0.22%)	435151 (18.29%)
$P_{Treestrace}$	7809563	942913 (12.07%)	1084790 (13.89%)	3822918 (48.95%)	765947 (9.81%)	1188528 (15.22%)

V. Discussion and Ethical Consideration

Flashroute [20] adopts the Doubletree strategy [21] for probe redundancy elimination in IPv4 network topology discovery. However, in this work, we have not yet considered its application for conserving measurement resources, mainly due to the challenge of incorporating the Doubletree strategy into the IPv6 domain. Specifically, this strategy relies on responses from next-hop devices to determine the probing of hops near the vantage point. This may not be easily achieved when the destination of the probing packet is inactive, given the sparse distribution of active IPv6 addresses. Furthermore, Flashroute achieves its high parallelism and preserves Doubletree's capabilities by assigning each target with a "destination control block," but this approach cannot be directly applied to the IPv6 realm due to the substantial storage requirements. Therefore, implementing Doubletree-based asynchronous probing techniques for IPv6 router interface discovery without seeds would necessitate significant modifications and improvements to the systems. We look forward to achieving this in the future.

We limited the probing rate to 100 Kpps to strictly ensure Internet citizenship, as suggested by Partridge and Allman [45]. We incorporated the website introducing our measurement purposes into the probe packets and removed any results related to entities that explicitly refused our disclosure requests.

VI. Conclusion

In this paper, we developed Treestrace, an asynchronous probing tool with high parallelism for IPv6 router interface discovery on a massive scale. Based on our novel weighted sampling technique, Treestrace achieves a dynamic probing strategy that automatically prioritizes the exploration of address spaces exhibiting a higher density of IPv6 router interface addresses. Real-world tests demonstrate that Treestrace outperforms state-of-the-art works in both seed-based and seedless measurements. By employing Treestrace, we discovered approximately 8 million IPv6 router interface addresses within several hours at a single vantage point. Its code is available at https://github.com/6Seeks/Treestrace.

Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Science Foundation of China (62072465), the Science and Technology Innovation Program of Hunan Province (Nos. 2022RC3061), and Postgraduate Research and Innovation Program of Hunan Province (ZC525Z042302). The corresponding author is Zhiping Cai.

REFERENCES

[1] Google, "IPv6 Adoption Statistics," 2023. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[2] W3Techs, "Usage Statistics of IPv6 for Websites," 2023. [Online]. Available: https://w3techs.com/technologies/details/ce-ipv6

[3] Akamai, "IPv6 Adoption Visualization," 2023. [Online]. Available: https://www.akamai.com/visualizations/state-of-the-internet-report/ipv6-adoption-visualization

[4] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer, "In the IP of the Beholder: Strategies for Active IPv6 Topology Discovery," in *Proc. IMC*, 2018, pp. 308–321.

[5] J. P. Rohrer, B. LaFever, and R. Beverly, "Empirical Study of Router IPv6 Interface Address Distributions," *IEEE Internet Computing*, vol. 20, no. 4, pp. 36–45, 2016.

[6] B. Hou, Z. Cai, K. Wu, J. Su, and Y. Xiong, "6Hit: A Reinforcement Learning-based Approach to Target Generation for Internet-wide IPv6 Scanning," in *Proc. INFOCOM*. IEEE, 2021, pp. 1–10.

[7] B. Hou, Z. Cai, K. Wu, T. Yang, and T. Zhou, "6Scan: A High-Efficiency Dynamic Internet-Wide IPv6 Scanner With Regional Encoding," *IEEE/ACM Transactions on Networking*, 2023.

[8] T. Yang, Z. Cai, B. Hou, and T. Zhou, "6Forest: An Ensemble Learning-based Approach to Target Generation for Internet-wide IPv6 Scanning," in *Proc. INFOCOM*. IEEE, 2022, pp. 1679–1688.

[9] O. Gasser and J. Zirngibl, "IPv6 Hitlist Service," 2023. [Online]. Available: https://ipv6hitlist.github.io/

[10] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists," in *Proc. IMC*. New York, NY, USA: ACM, 2018.

[11] E. C. Rye and R. Beverly, "Discovering the IPv6 Network Periphery," in *Proc. PAM*. Springer, 2020, pp. 3–18.

[12] K. Fukuda and J. Heidemann, "Who Knocks at the IPv6 Door? Detecting IPv6 Scanning," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 231–237.

[13] J. Czyz, M. Luckie, M. Allman, M. Bailey *et al.*, "Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy," in *Proc. NDSS*, 2016.

[14] L. Pan, J. Yang, L. He, Z. Wang, L. Nie, G. Song, and Y. Liu, "Your Router is My Prober: Measuring IPv6 Networks via ICMP Rate Limiting Side Channels," in *Proc. NDSS*, 2023.

[15] M. Luckie and R. Beverly, "The Impact of Router Outages on the AS-level Internet," in *Proc. SIGCOMM*, 2017, pp. 488–501.

[16] D. Plonka and A. Berger, "KIP: A Measured Approach to IPv6 Address Anonymization," *arXiv preprint arXiv:1707.03900*, 2017.

[17] T. V. Doan, V. Bajpai, and S. Crawford, "A Longitudinal View of Netflix: Content Delivery over IPv6 and Content Cache Deployments," in *Proc. INFOCOM*. IEEE, 2020, pp. 1073–1082.

[18] D. Plonka and A. Berger, "Temporal and Spatial Classification of Active IPv6 Addresses," in *Proc. IMC*, 2015, pp. 509–522.

[19] R. Beverly, "Yarrp'ing the Internet: Randomized high-speed active topology discovery," in *Proc. IMC*, 2016, pp. 413–420.

[20] Y. Huang, M. Rabinovich, and R. Al-Dalky, "Flashroute: Efficient Traceroute on a Massive Scale," in *Proc. IMC*, 2020, pp. 443–455.

[21] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. SIGMETRICS*, 2005, pp. 327–338.

[22] X. Li, B. Liu, X. Zheng, H. Duan, Q. Li, and Y. Huang, "Fast IPv6 Network Periphery Discovery and Security Implications," in *Proc. DSN*. IEEE, 2021, pp. 88–100.

[23] U. of Oregon, "Route views project," 2023. [Online]. Available: https://www.routeviews.org/routeviews/

[24] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Internet Requests for Comments, RFC Editor, STD 86, July 2017.

[25] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," Internet Requests for Comments, RFC Editor, RFC 4291, February 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4291.txt

[26] M. Boucadair, A. Petrescu, and F. Baker, "IPv6 Prefix Length Recommendation for Forwarding," Internet Requests for Comments, RFC Editor, BCP 198, July 2015.

[27] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," Internet Requests for Comments, RFC Editor, BCP 122, August 2006, http://www.rfc-editor.org/rfc/rfc4632.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4632.txt

[28] B. Carpenter, T. Chown, F. Gont, S. Jiang, A. Petrescu, and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing," Internet Requests for Comments, RFC Editor, RFC 7421, January 2015.

[29] A. APNIC and N. RIPE, "IPv6 Address Allocation and Assignment Policy," 2020. [Online]. Available: https://www.ripe.net/publications/docs/ripe-738

[30] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," Internet Requests for Comments, RFC Editor, RFC 4862, September 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4862.txt

[31] T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," Internet Requests for Comments, RFC Editor, RFC 4941, September 2007.

[32] T. Mrugalski, M. Siodelski, B. Volz, A. Yourtchenko, M. Richardson, S. Jiang, T. Lemon, and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," Internet Requests for Comments, RFC Editor, RFC 8415, November 2018.

[33] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Internet Requests for Comments, RFC Editor, RFC 4443, March 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4443.txt

[34] B. Hou, T. Yang, Z. Cai, K. Wu, and T. Zhou, "Search in the Expanse: Towards Active and Global IPv6 Hitlists," in *Proc. INFOCOM*. IEEE, 2023.

[35] CAIDA, "The CAIDA Internet Topology Data Kit," 2023. [Online]. Available: http://www.caida.org/data/internet-topology-data-kit

[36] RIPE, "RIPE Atlas," 2023. [Online]. Available: https://atlas.ripe.net/

[37] M. Luckie, "Scamper: A Scalable and Extensible Packet Prober for Active Measurement of the Internet," in *Proc. IMC*, 2010, pp. 239–245.

[38] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *Proc. USENIX Security*, vol. 8, 2013, pp. 47–53.

[39] H. Yuchen and M. Maxime, "FlashRoute6 (IPv6 Compatible)," 2023. [Online]. Available: https://github.com/lambdahuang/FlashRoute

[40] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial Optimization*. Springer, 2011, vol. 1.

[41] A. Moffat, "Huffman Coding," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–35, 2019.

[42] J. Zirngibl, L. Steger, P. Sattler, O. Gasser, and G. Carle, "Rusty Clusters? Dusting an IPv6 Research Foundation," in *Proc. IMC*. New York, NY, USA: ACM, 2022.

[43] L. Hendriks, "zesplot: IPv6 visualisation based on squarified treemaps," 2023. [Online]. Available: https://github.com/zesplot/zesplot

[44] E. Rye, R. Beverly, and K. C. Claffy, "Follow the Scent: Defeating IPv6 Prefix Rotation Privacy," in *Proc. IMC*, 2021, pp. 739–752.

[45] C. Partridge and M. Allman, "Ethical Considerations in Network Measurement Papers," *Communications of the ACM*, vol. 59, no. 10, pp. 58–64, 2016.