6Scan: A High-Efficiency Dynamic Internet-Wide IPv6 Scanner With Regional Encoding

Bingnan Hou, Zhiping Cai, Kui Wu, *Senior Member, IEEE*, Tao Yang, and Tongqing Zhou

***Abstract*—Efficient Internet-wide scanning plays a vital role in network measurement and cybersecurity analysis. While Internet-wide IPv4 scanning is a solved problem, Internet-wide scanning for IPv6 is still a mission yet to be accomplished due to its vast address space. To tackle this challenge, IPv6 scanning generally needs to use pre-defined seed addresses to guide further IPv6 scanning directions. Under this general principle, various solutions have been developed, but all suffer from two primary pitfalls, low hit rate and low probing speed, caused by the inherent sparse distribution of active IPv6 addresses and the high computational complexity of the search algorithms, respectively. We develop 6Scan, a novel asynchronous IPv6 scanner that effectively addresses the above two problems. To increase the hit rate, 6Scan infers the promising search directions by encoding the regional identifiers of the target addresses within the probing packets and recording the regional activities from the asynchronously arrived replies. It then dynamically adjusts the search directions according to the scanning result of the previous steps. To speed up the search algorithm, 6Scan leverages the regional identifier encoding to quickly adjust search direction without excessive computation. Real-world experiments over the IPv6 Internet in a billion-scale probing budget show that compared with the state-of-the-art solutions, on average 6Scan can discover 6% more active addresses with nearly the same scanning time.**

***Index Terms*—IPv6, network measurement, internet-wide scanning, alias prefix resolution.**

I. Introduction

EFFICIENT Internet-wide scanning is an essential building block in network measurement and cybersecurity analysis. In the past, asynchronous scanning tools like ZMap [1] and Masscan [2] have drastically enhanced our capability of conducting Internet-wide network surveys, e.g., topology discovery [3], [4], IP address analysis [5], [6], and geolocation [7]. Network device search engines like Shodan [8] and Censys [9] can acquire Internet-wide asset data for evaluating network security, discovering vulnerabilities, and tracking remediations [10]. Nevertheless, these tools are not effective when applied for IPv6 Internet, because the huge address space of IPv6 renders comprehensive scan nearly impossible. Due to this reason, developing efficient worldwide network security analysis for IPv6 is still an open challenge.

To answer the above challenge, people have concluded that a brute-force scan in IPv6 is infeasible, and a solution for efficiently discovering active addresses should be the first critical step. In this regard, previous works usually utilize existing high-speed scanners (e.g., ZMapv6 [1]) for probing and then improve the hit rate by generating address targets that are likely to be active. Generally, they use a set of known active addresses, also called seeds, to discover more new active addresses in the IPv6 address space. The common belief is that the seeds can provide us with useful information in the search for active addresses. However, these searching strategies either face the problem of low hit rate as the seeds may not well match the distribution of active addresses or low probing speed due to intensive computations of the search algorithms. Thus, we are motivated to develop a new ping-like scanning tool specifically for IPv6 active address search, with the goal of improving both hit rate and probing speed.

We propose 6Scan, a dynamic scanning tool that initially utilizes the information provided by the seeds and then learns the better search directions according to the feedback during the scanning process. The main idea is that even if the initial seeds misalign the "true" distribution of IPv6 active addresses, the feedback from existing search results reflects the real picture better. By gradually augmenting our knowledge regarding the actual IPv6 active address distribution, we can adjust the search towards more promising directions.

6Scan concretes the above idea by asynchronously collecting regional activity information from the interaction with the real network. In particular, it first utilizes the seeds to partition the IPv6 address space and then takes the regional activity information embedded in the scanning result of previous steps to guide the subsequent search directions. By dynamically adjusting search direction, 6Scan can achieve a much higher hit rate than existing static scanning approaches [11], [12].

To improve probing speed, 6Scan abandons high complexity algorithms used in existing methods [13] and instead only uses

Manuscript received 1 December 2021; revised 21 May 2022 and 2 December 2022; accepted 21 December 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Balasubramanian. Date of publication 9 January 2023; date of current version 18 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62072465, Grant 62172155, Grant 62102425, and Grant 62102429; in part by the Open Foundation of Henan Key Laboratory of Cyberspace Situation Awareness under Grant HNTS2022006; and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2022RC3061 and Grant 2021RC2071. *(Corresponding authors: Zhiping Cai; Tongqing Zhou.)*

Bingnan Hou is with the College of Computer, National University of Defense Technology, Changsha 410073, China, and also with the Henan Key Laboratory of Cyberspace Situation Awareness, Zhengzhou 450001, China (e-mail: houbingnan19@nudt.edu.cn).

Zhiping Cai, Tao Yang, and Tongqing Zhou are with the College of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: zpcai@nudt.edu.cn; yangtao97@nudt.edu.cn; zhoutongqing@nudt.edu.cn).

Kui Wu is with the Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: wkui@uvic.ca).

Digital Object Identifier 10.1109/TNET.2023.3233953

simple counting to adjust its search directions swiftly. 6Scan achieves its efficiency by design to be asynchronous with state into the probing process, which is different from simple synchronous approaches (e.g., [13]) and other asynchronous stateless scanning tools (e.g., [1]). The state is carefully designed to preserve high probing speed yet enable dynamic probing strategies that increase the number of probes that reach the active hosts. To be more specific, each probing packet sent from 6Scan is encoded with the identifier of the address subspace so that 6Scan can directly obtain the activity of the subspace through a simple regional activity counter, which is updated with the replies arriving asynchronously. This regional encoding scheme dramatically reduces the computation time in adjusting the search directions. The above two salient features (i.e., asynchronous model and regional encoding) make 6Scan, to our knowledge, the most efficient IPv6 address scanner so far.

Overall, 6Scan is a high-efficiency prober for Internet-wide IPv6 scanning, which not only takes inspiration from the well-known stateless and randomized high-speed scanners (e.g., ZMap [1] and Yarrp [3]) but also integrates the idea of reinforcement learning to dynamically adjust the search directions to subspaces with higher activity (i.e., higher rewards) [13]. 6Scan is written in C++, is portable to UNIX-like platforms. We released our tool along with the scan results of discovered addresses and alias prefixes at https://hbn1987.github.io/HMap6.github.io/.

The main contributions of the paper are as follows:

- We develop a novel asynchronous scanning tool, 6Scan, for efficient Internet-wide IPv6 active address search. 6Scan is resilient to the quality of initial seed addresses obtained in various ways, e.g., using public seedsets or heuristically collecting seeds under the announced BGP prefixes. Equipped with the seed addresses, it can quickly generate a large number of potential active addresses for subsequent scanning. This capability lays a foundation for other follow-on research such as topology discovery and security analysis in the IPv6 Internet.
- 6Scan takes both hit rate and probing speed into account and tries to achieve a high hit rate with fast adjustment of search directions guided with straightforward calculation. Leveraging the idea of reinforcement learning and the regional identifier encoding, 6Scan quickly adjusts the search directions towards promising regions that contain more active addresses. Real-world experiments over the IPv6 Internet in a billion-scale probing budget[1] show that compared with the state-of-the-art solutions, 6Scan can discover 6% more active addresses with nearly the same scanning time.
- 6Scan is modular designed as a ping-like probing framework, which we make publicly available to allow other researchers to develop other active search strategy easily. Over this framework, we have optimized the performance of the existing mainstream search strategies (i.e., [12], [13], [14]) by adapting them to large-scale scanning to facilitate more objective comparison and analysis.
- We propose a novel heuristic method for active address (seed) discovery at the regional level (e.g., country and AS), which combines a simple yet efficient de-aliasing algorithm in the process of active search to get more valuable seeds in various prefixes. Experimental results show that the heuristic regional seed discovery method can find seeds in various regions that are different from the existing published open-source seed list. The newly-found seeds greatly help future research relevant to the regional activity search in IPv6 networks.
- We run 6Scan at almost 100K packets per second (pps) using the largest seedset as we know to discover more than 16 million active IPv6 addresses within a few hours. We also survey the IPv6 addressing patterns of the scanning results and make a detailed comparison and analysis of the addressing patterns discovered by each searching strategy.

The remainder of the paper is organized as follows. Section II summarizes the background and related work on IPv6 Internet-wide scanning. Section III explores the considerations for IPv6 scanning using target generation. Section IV presents the system design of 6Scan. Section V introduces our novel heuristic seed collection method. Section VI evaluates our seed collection method and compares the performance of 6Scan with representative alternatives. Section VII concludes the paper.

II. Background and Related Work

A. Target Generation Algorithms

Prior works on IPv6 active search mainly design algorithms that utilize seeds to generate address targets to scan. They assumed that the known active addresses provide information on the use of addressing schemes, that is, the seed information is helpful in discovering more new addresses. So far, related researches have exploited both the statistical information and the structural information in the seeds. Accordingly, we group related research into two classes: address pattern-based and density-driven methods.

In the first class, the address pattern extracted from the statistic information of seeds is exploited to guide the target address generation [11], [15], [16]. Ullrich et al. [15] utilized a recursive algorithm to greedily determine the bit by reaching the most addresses in the seedset in each recursion. Foremski et al. [11] introduced Entropy/IP, an algorithm for learning patterns from seeds, which utilizes empirical entropy to group adjacent nybbles of IPv6 addresses into segments and uses a Bayesian network to model the statistical dependencies between values of different segments. The learned statistical model is used to generate target addresses for scanning. Cui et al. [16] used a deep learning approach to target generation which utilizes multiple generators to learn IPv6 addressing patterns determined by the seeds. However, these address pattern-based search strategies are not suitable for large-scale scanning in real networks. When the number of seed addresses is small, these algorithms can not extract effective address patterns from the seeds, resulting in a low hit rate. But when the number of seeds is large enough, the scanning efficiency is reduced due to the high computational

[1]Probing budget is defined as the number of allowed probing attempts.

complexity of these algorithms that have high overhead in training time.

In the second class, the density of seeds in the address space is mainly used to determine the scanning area [12], [14], [17], [18]. Murdock et al. [12] proposed 6Gen, which assumes that the address space with high-density seeds is more likely to have undiscovered active addresses. 6Gen greedily expands each seed as a center of each cluster to generate the target addresses by maintaining the maximal seed density and the minimal scale. Liu et al. [14] proposed 6Tree, which takes advantage of a space tree formed from seeds' structure to divide the IPv6 address space. 6Tree calculates the density of active nodes on the space tree according to the known active addresses recorded on the nodes. It then generates target addresses based on the density of active nodes. These two density-driven target generation algorithms are static scanning approaches, i.e., the seed density in the address space directly determines the subsequent scanning areas. Hou et al. [13] introduced a dynamic adjustment method into the scanning process, which utilizes a reinforcement learning algorithm to make the preorder scanning results guide the subsequent search directions and enhance the interaction between the scanning process and the real networks.

Different from the above ping-based scanning approaches that send one packet per IP with a fixed TTL (e.g., 128 hops), Rye et al. [19] used a high-speed traceroute-based prober, namely Yarrpv6 [4] for active address discovery, which can find the routing interfaces on the traceroute paths. Recently, FlashRoute [20] has been proposed as a more efficient traceroute tool for network topology mapping. Note that the key idea of 6Scan and FlashRoute is similar that not all state in the scanning process is evil as long as the prober can fit the required footprint into its memory. Thus, both 6Scan and FlashRoute retain the innovation of Yarrp in encoding all necessary fingerprint in probe headers to enable high probing speed but keep a limited amount of state to facilitate more intelligent probing strategies. The only difference is that 6Scan retains the regional active density information in the prober's memory while FlashRoute preserves previously discovered interface inspired by Doubletree [21]. However, the traceroute-like approach needs to send more packets per IP with varying TTLs (i.e., IPs $\times$ TTLs). Therefore it may not be suitable for scanning aimed at efficient active host discovery.

B. Alias Prefix Resolution

Alias resolution is the process of determining if IP addresses are assigned to the same physical device (e.g., router and server) [22]. Aliased prefixes, namely, prefixes under which each possible IP address replies to queries, are very common in IPv6. Through the IP_FREEBIND option in Linux, this feature can be easily deployed in CDNs [23]. Yet, alias resolution poses a challenge in IPv6 measurement as it can easily contribute vast numbers of addresses that map to the same server (e.g., enumerating a /48 prefix can add 2^{80} addresses). Note that this feature has a severe impact on dynamic target generation algorithms (e.g., [13]), as they will get more feedback when scanning the aliased areas, making the following-on scanning fall into the "alias trap".

Several IPv6 alias resolution techniques are proposed [24], [25], [26], [27], [28], [29]. Albakour et al. [24] exploited SNMPv3 protocol for alias resolution by analyzing the engine ID response to unsolicited and unauthenticated SNMPv3 requests recently. This protocol-based alias resolution approach can achieve high accuracy yet is limited by the requirement that the router must support the specific protocol. TBT [25] and Speedtrap [26] leveraged the ability to induce fragmented IPv6 responses from router interfaces to perform alias resolution according to the characteristics of the IPID values. Limited Ltd [28] sends exponentially increasing ICMPv6 packets to the candidate alias pairs and uses the similarity of packet loss rates displayed by the devices to determine the alias addresses. Although these three fingerprint-based alias resolution methods have high accuracy, they need to send a large number of packets to the devices, which will bring too much burden on the network. APPLE [29] filters potential router aliases seen in traceroute by comparing the reply path length (i.e., TTL) from each address. However, APPLE has a high deployment cost since it requires distributed set of vantage points to measure the path length to the potential alias addresses. Padmanabhan et al. [27] proposed AUv6, an alias resolution method based on prefix detection. However, AUv6 can only detect the /126 prefixes that cannot resolve the prefixes of other lengths.

Gasser et al. [23] proposed APD, a prefix detection approach especially suitable for large-scale alias prefix resolution. APD is based on the observation originally from [12] that a randomly selected IP address in the vast IPv6 space is unlikely to respond. So it generates random addresses under the prefix in a regular way, and if all of these addresses respond then the prefix is judged to be an alias prefix. By utilizing APD, Gasser published a wide-range alias prefix list, which has been regarded as the ground-truth by several previous works [16], [19].

C. Hitlist and Seeds

The hitlist catalogues active IPv6 addresses on the Internet and is commonly used as the seed addresses for the target generation algorithms. Notable efforts leveraged active measurements (e.g., reverse DNS zone walking [30], [31] and iterative probing subnets [19]), passive techniques (e.g., from BGP updates [32] and traffic capture [33]), as well as a number of passive and active sources [33], [34].

The public hitlist from Gasser et al. [23] collects a wide range and a large number of IPv6 addresses from multiple sources (e.g., crowdsourcing platform [35], Bitnodes API [36], RIPE IPMap [37], Scamper [38]). In addition, Gasser's weekly published hitlist excludes alias addresses to reduce the possibility that a target generation algorithm over-scans the alias areas. Gasser's hitlist plays a vital role in promoting the IPv6 measurement community that nearly almost all the proposed target generation algorithms extract addresses from their hitlist to form the seedset.

However, there are some regional imbalances in the collection of Gasser's hitlist. For example, some regions or prefixes have too many addresses and some have almost none. In this work, we propose a heuristic active search algorithm for regional seed discovery which also takes advantage of an efficient alias resolution method inspired by the APD algorithm [23] to avoid introducing alias addresses in the process of regional seed collection to improve the quality of seeds.

III. IPv6 Scanning Considerations

While the current network speeds and computational power can support researchers to make Internet-wide IPv4 scan in minutes using a brute-force manner [1], [2], the IPv6 address space is too ample for such scan methods. To effectively discover active addresses, IPv6 scanners need to be more intelligent to increase the hit rate with a high probing speed. In this section, we explore how to utilize existing information (i.e., seeds) and design effective search strategies to improve the scanning efficiency.

A. Seed Addresses and Space Partition

An IPv6 address can be represented as a hexadecimal string with 32 nybbles. Alternatively, we can consider an IPv6 address as a vector in a 32-dimensional space (i.e., address space). The value in each dimension of the vector is an integer in $[0, 15]$ (i.e., the value range of a nybble). Following the convention, we use the wildcard symbol "*" to denote a nybble which can be any value in $[0, 15]$. As mentioned in Section II-A, density-driven approaches can partition the address space $\mathcal{X}$ into regions for the probers to scan. In the following, we use regions and subspaces interchangeably. Since the density of active addresses in each region is initially unknown, density-driven approaches should find a way to learn the address density in each region and generate targets in high-density regions to increase the hit rate. To this end, they assume that the seeds are random samples from the set of all active addresses $\mathcal{A}$ (in the address space $\mathcal{X}$) and explore the hidden information in the seeds to facilitate the partitioning of $\mathcal{X}$.

Existing researches mainly adopt a hierarchical clustering algorithm to group seeds into a hierarchical "tree" of clusters, each cluster representing a region in the address space $\mathcal{X}$. This clustering algorithm can use either agglomerative or divisive clustering, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) approach.

Examples of agglomerative and divisive clustering of seeds are shown in Fig. 1 (a) and (b), respectively. As shown in Fig. 1 (a), the agglomerative hierarchical clustering (AHC) algorithm starts by instantiating with a cluster for each seed ($c_i, i \in [1, 5]$). In each iteration, it calculates the vector distance between clusters and merges two clusters with the minimum distance. One representative method using AHC algorithm is 6Gen [12]. It is worth noting that different from the general AHC algorithm, 6Gen does not explicitly merge similar clusters. Instead, each cluster grows independently, and it allows a seed address to belong to multiple clusters. This feature makes AHC accurately calculate the number of seeds in the clusters, but it increases the time complexity of the algorithm to $O(n^3)$, where n denotes the number of seeds. Unfortunately, given a large seedset, the long space partition time will certainly decrease the scanning efficiency.

The divisive hierarchical clustering (DHC) algorithm adopts a top-down clustering approach, as shown in Fig. 1 (b). It starts with a single cluster containing all seeds. After each iteration, it splits the clusters from the left-most variable dimension (i.e., dimension with value of "*"). It stops when each seed is in its own singleton cluster or the cluster can no longer split (i.e. only one variable dimension left). The DHC algorithm has been proved to be superior compared with AHC by many previous works [13], [14], [33]. First, its computational complexity is $O(nlogn)$, which scales better for a large seedset. Second, in the top-down splitting process of the algorithm, the clusters gradually shrink until all the seeds are contained tightly. As Fig. 1 (b) shows, the four address subspaces (i.e., 2001::000*, 2002::00**, 2003::001*, and 2003::002*) contain all seeds with only 304 $(= 16 + 16^2 + 16 + 16)$ target addresses. However, the subspace (i.e., 200*::00**) that covers all seeds formed by AHC generates 4096 $(= 16^3)$ targets. In other words, the DHC algorithm makes the scanning interval contain all seeds as much as possible in the case of a small probing budget, so it makes better use of the seed addresses. Due to this reason, 6Scan adopts the DHC algorithm. Note that there is no significant advantage or disadvantage between these two space partition algorithms in the case of a large probing budget, because the scanning interval can contain all seeds no matter whether we use AHC or DHC. A toy example of space partition with 6Scan will be disclosed in Section IV-B.

B. Seed Density and Activity Density

A density-driven approach treats the seeds as independently and identically distributed (iid) random samples from the set of active addresses $\mathcal{A}$ in the address space $\mathcal{X}$. As such, the regions of address space with the highest density of active addresses will likely have the highest density in seeds. In other words, the activity density (i.e., the number of active addresses) in each region is estimated by the seed density in this region. For example, 6Gen [12] and 6Tree [14] determine the sequence of regions to be scanned according to the seed density in the regions.

One main pitfall of these static density-driven scanning approaches is that the seed density in a region may not align well with the activity density in the region. In practice, samples may have noise and may be biased. If we take the fitting degree between the distribution of seeds and the distribution of real active addresses as the only basis for target generation, the density-driven method would depend too much on the quality of seeds. To avoid over-reliance on seeds, 6Hit [13] takes advantage of a bandit algorithm [39] to introduce the action-reward mechanism into the scanning process, and dynamically adjusts the search directions according to the evaluative feedback. Although the reinforcement learning-based search strategy can improve the hit rate to some extent, especially in the case of poor seed quality, its complex algorithm may lead to a sharp decline in probing speed as it needs to continuously calculate the region to which the replies belong. Inspired by 6Hit, 6Scan also adopts a dynamic search strategy but significantly reduces the complexity of the search algorithm to achieve a high probing speed.

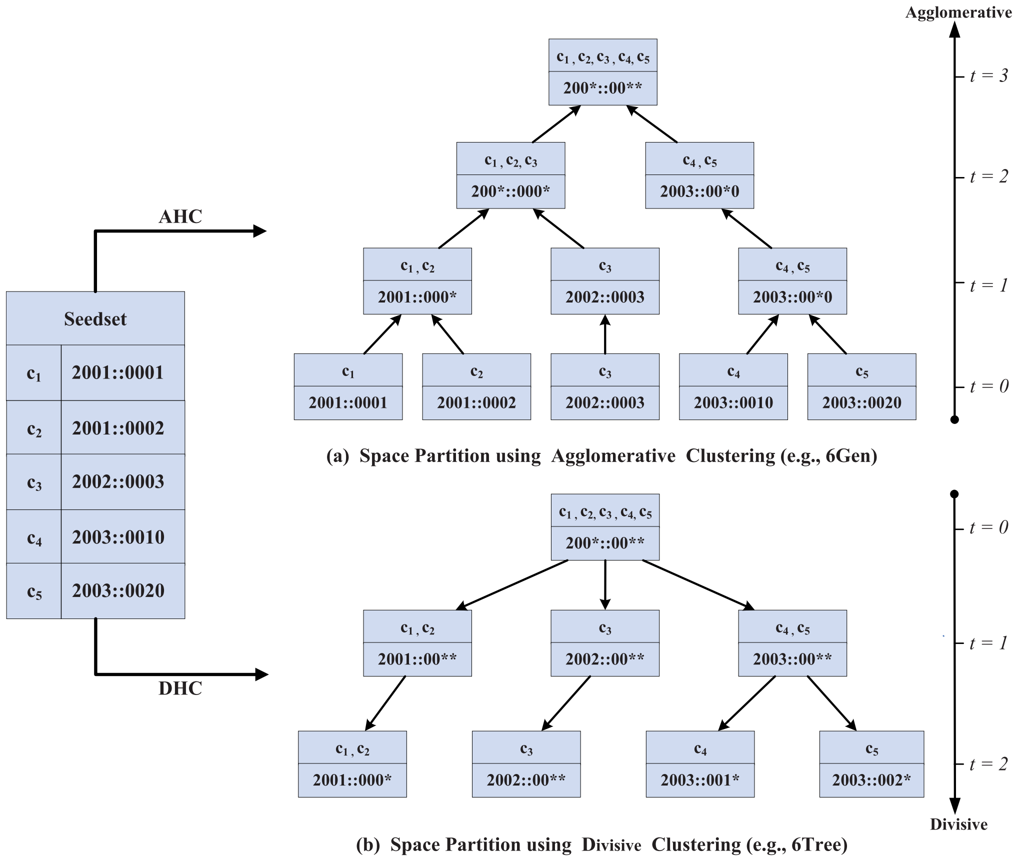

Fig. 1. Examples of space partition using hierarchical clustering.

C. Hit Rate and Scanning Efficiency

To handle the large IPv6 address space and limit the scan in a reasonable amount, prior works usually incorporate the notion of a probe budget that specifies the total number of packets that a scanner can send. They focus on how to utilize the seeds to improve the hit rate, i.e., $\frac{\mathcal{N}}{\mathcal{B}}$, where $\mathcal{N}$ and $\mathcal{B}$ denote the newly discovered de-aliased addresses and the total number of budget, respectively. However, improving the hit rate alone does not necessarily mean the increase of the scanning efficiency, because we also need to consider the scanning time. For example, the deep learning-based approach 6Gan [16] achieves a very impressive hit rate within a small budget but requires a long training time. Obviously, this deep learning-based approach is not suitable for large-scale and fast network scanning.

In this work, we aim to design a truly efficient prober (w.r.t. both hit rate and scanning time) for Internet-wide IPv6 scanning. Our goal is to find the most newly discovered active addresses $\mathcal{N}$ given a seedset $\mathcal{C}$ within a (large) fixed budget $\mathcal{B}$ as quickly as possible, i.e., $Maximize\{\frac{\mathcal{N}}{\mathcal{T}}\}$, where $\mathcal{T}$ denotes the total time of the scanning.

IV. 6Scan Design

We first present the system overview of 6Scan and then describe the details of its key technical components: 1) **address space partition**, which utilizes the structure information of the seeds to form the candidate scanning subspaces; 2) **dynamic target generation**, which uses the previous scanning results to guide the subsequent target generation; and 3) **regional encoding technique**, which encodes the regional information of the target into the probes such that the regional activity can be obtained directly from the receiving thread.

A. System Overview

6Scan is an open-source network scanner optimized for efficiently performing IPv6-wide active search. It uses a modular design to support dynamic search with various types of probes (i.e., ICMPv6, TCP, and UDP) and utilizes independent sending and receiving threads to conduct asynchronous scanning. The architecture of 6Scan is shown in Fig. 2, in which ellipses indicate the input (e.g., hitlist and announced prefixes) or output (e.g., scanning results) data, the rectangles indicate the modules of the system, and the arrows represent the data flow or control operation between modules.

Initially, 6Scan downloads public domain information, e.g., Gasser's hitlist from its service [40] and the announced prefixes, using RIPEstat data API [41]. 6Scan can form the seedset in two ways. One is to pre-scan Gasser's hitlist and use the response addresses to form the seedset. The other is a region-level seed collection using a heuristic strategy to scan the announced prefixes. Note that both ways of seed collection do a prior scan to ensure that the collected seeds are active. Usually, the pre-scanning can exclude a large number of unresponsive addresses from Gasser's hitlist [34]. Unresponsive addresses are mainly due to several reasons: Firstly, the Gasser's hitlist is a union of reachable addresses from different protocols (e.g., ICMPv6, TCP, and UDP) while

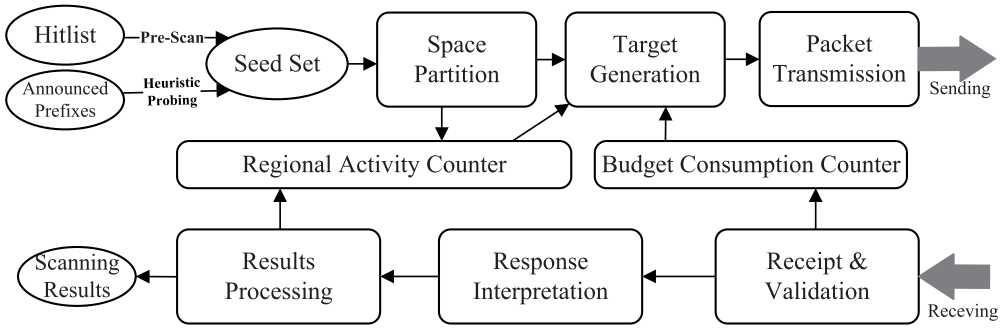

Fig. 2. 6Scan architecture.

the pre-scanning produces the seedset that only responds to one protocol at a time. Secondly, some addresses acquired by passive measurement do not respond to active detection. Thirdly, different scan origins affect the results of scans due to permanent and temporary blocking, packet loss, geographic biases, and other reasons [42].

Then, modular space partition utilizes the seed to divide the entire IPv6 address space $\mathcal{X}$ using the DHC algorithm to form the "space tree". Next, the target generation module generates target addresses by traversing the space tree using a bottom-up approach, i.e., generating targets from the underlying small address space at first. Meanwhile, 6Scan utilizes a regional activity counter to estimate the return of active addresses in each region. Whenever there is a reply, 6Scan updates the regional activity counter and then adjusts the future probing packets accordingly. This asynchronous and dynamic scan-update-adjust process cycle continues until the total number of probes reaches the budget limit.

In addition, 6Scan supports multiple kinds of probes (e.g., TCPv6 SYN and ICMPv6 echo) and encodes all important states (e.g., scanner fingerprint and regional identifier) obtained from packet transmission and response interpretation components.

B. Address Space Partition

The partition of the address space $\mathcal{X}$ is realized by utilizing the DHC algorithm to construct a tree structure named the "space tree." The tree is built from the root to leaves by hierarchically dividing the 32-dimensional space of $\mathcal{X}$.

The pseudocode is shown in Algorithm 1. At the beginning of the space tree construction, the root node contains all address vectors in $\mathcal{X}$, i.e., the 32 dimensions of the root are all marked with "*". A dimension marked with "*" is called variable dimension, while a dimension marked with a constant (i.e., $[0 - f]$) is called constant dimension. Since the space tree is constructed according to the structural information embedded in the seed addresses, each node on the tree contains two attributes: (1) the assigned dimension ($assignedDimension$) which is used to represent the address subspace allocated to the node, and (2) the assigned seed ($assignedSeed$) which is used to finally determine the $assignedDimension$ (i.e., the subspace) associated with the node. In the process of cluster splitting, DHC performs seeds segmentation according to whether the leftmost variable dimension has the same value, that is, the addresses associated with the same child node have the same value on their assigned dimensions.

Fig. 3 shows an example of building a space tree with 500 seed addresses. The final tree contains 65 nodes in total, and we only draw some of the nodes for a clear illustration. It can be seen that 6Scan applies a top-down clustering method. Advancing from the root to leaf nodes, the number of variable dimensions is gradually reduced. As shown in the figure, the root node d_0 has a child node d_1. The $assignedDimension$ of d_1 represents the address subspace whose first 6 dimensions are 200112 and the remaining dimensions are variable dimensions (marked with "*").

C. Dynamic Target Generation

After address space partition, we need to determine an appropriate allocation of the probes to achieve a high hit rate with a high probing speed. Like 6Hit [13], we adopt a dynamic approach for target generation that evaluates the feedback from the replies received so far to adjust the probing directions. We utilize the dynamic search strategy based on the consideration that the seed density may largely differ from a region's activity density. Hence, we should not rely on seed density too much. Instead, we should directly use the previous scanning results to guide the subsequent probing directions. However, 6Hit [13] only focuses on improving the hit rate. It distinguishes the address subspaces of every reply, then evaluates the feedback of each subspace, and finally adjusts the search directions in each scan-update-adjust iteration. In contrast, 6Scan takes both the hit rate and probing speed into considerations. It encodes the regional identifier into the packet header and designs a regional activity counter to record the returned active addresses in each region. When the reply is received by the receiving thread, the results processing module decodes the payload of the packet and updates the regional activity counter. 6Scan then queries the regional activity counter and adjusts the search direction accordingly. By simply checking the regional activity counter, this asynchronous process greatly reduces the time of search direction adjustments. In this way, 6Scan can maintain a higher probing speed compared with 6Hit [13].

The dynamic target generation algorithm is shown in Algorithm 2. The algorithm takes as input a partitioned address space list $X = \{X_1, \ldots, X_n\}$ and the probing budget $\mathcal{B}$. In fact, every X_i has a bijective relationship with every node on the space tree. The size of X_i can be calculated by the number of variable dimensions

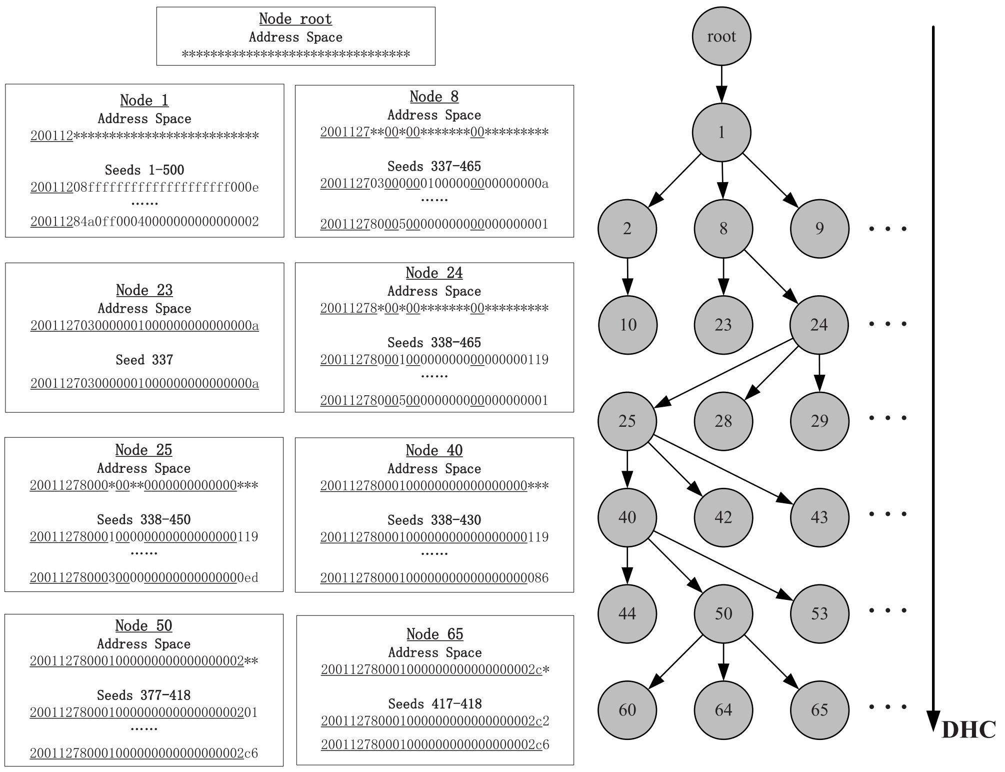

Fig. 3. Toy example of space tree generation. It generates a 65-node tree (partially shown) from 500 seed addresses.

(i.e., $|unassignedDimension|$) of its corresponding leaf node. Then, the algorithm starts to traverse the nodes in the space tree from bottom to top, and generates all the addresses contained in the corresponding node as the targets for probing. Note that a packet sent by 6Scan is encoded with its regional identifier (i.e., node number) such that the receiving thread can continuously update the regional activity counter from the replies. The activity of the scanned nodes will be synchronized to their parent nodes, i.e., the scanning order of subsequent nodes is determined by the activities of all its child nodes. Finally, the target generation will be prioritized in regions with high activities until the budget limit is reached.

D. Regional Encoding Technique

6Scan leverages the idea of packet encoding to conduct asynchronous scanning from Yarrpv6 [4] and ZMap [1]. The difference is that 6Scan adds the encoding of the regional identifier (i.e., the node number in the space tree) in the packet payload to support dynamic scanning.

Fig. 4 depicts 6Scan state encoding within the probes it sends. The 6Scan payload of 32B is added after the IPv6 and transport headers. The 6Scan payload consists of a 4B scanner fingerprint and 1B instance ID to ensure that the received packets are indeed responses to 6Scan probes. A single byte encodes the originating TTL (e.g., 128 hops) that the difference with the “hop limit” gives the number of hops to the destination. Four bytes encode the probe’s timestamp to permit round-trip-time (RTT) computation, and four bytes encode the regional identifier to update the regional activity from the reply. The regional encoding aims at the efficient mapping of a reply and its corresponding region/node. Directly

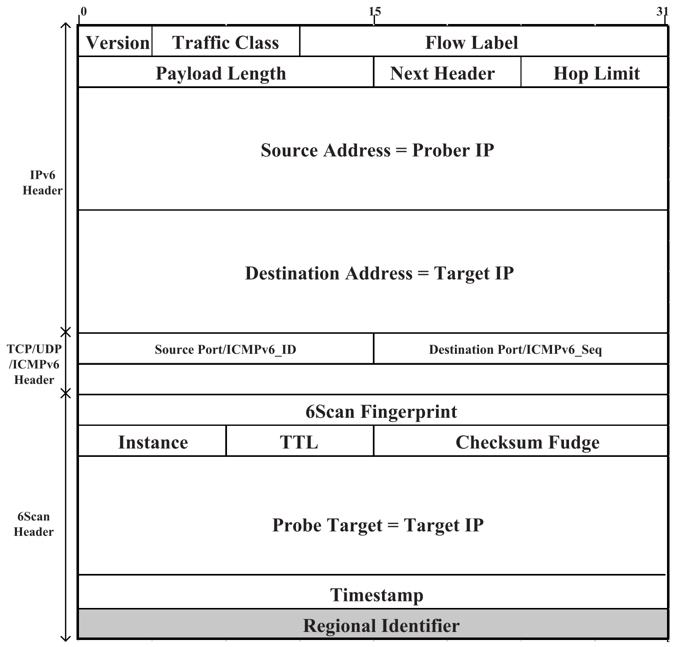

Fig. 4. States encoding in packet header of 6Scan.

matching replies to regions/nodes on the receiving thread without regional encoding is also an option, but intuitively this scheme takes up more memory that needs to store the strings of subspaces rather than merely maintaining a list of region/node numbers, and more CPU computation overhead for string comparison of source addresses to subspaces rather than simply looking at the decoded regional identifier. As with Yarrpv6 [4], the two bytes of “fudge” within the payload correct the checksum such that the IPv6 and transport headers

Algorithm 1 Address Space Partition

```
Input: The seedset $\mathcal{C}$;
Output: The root node of the space tree $d_0$;
1: $d_0 = InitializeRoot()$;   ▷ Initializes the root node, specifying that its child is $d_1$. The address space of root is the full space.
2: $d_1 = CreateNode(\mathcal{C})$;
3: $DHC(d_1)$;
4: return $d_0$;
5: function $CreateNode(C)$   ▷ Creates a node and assigns the seeds $C$ and dimensions to it.
6:     $newNode.assignedSeed = C$;
7:     $newNode.assignedDimension = newNode.parentNode.assignedDimension$;
8:     return $newNode$;
9: end function
10: function $DHC(node)$
11:     if $|node.assignedDimension| \geq 31$ then   ▷ $|node.assignedDimension|$ indicates the number of constant dimensions.
12:         return ;
13:     end if
14:     for $\delta = 1$ to 32 do
15:         if $\forall c_i, c_j \in node.assignedSeed,\ c_i[\delta] == c_j[\delta]$ then
16:             $node.assignedDimension.Append(\delta)$;
17:         end if
18:     end for
19:     for $i = 1$ to 32 do
20:         if $i \notin node.assignedDimension$ then
21:             $\delta^* = i$;
22:             Break;
23:         end if
24:     end for
25:     $subsequences = Parition(node.assignedSeed, \delta^*)$;   ▷ Splits out subsequences of the assigned seeds on node where the seed vectors have the same value in dimension $\delta^*$.
26:     for $\sigma \in subsequences$ do
27:         $newNode = CreateNode(\sigma)$;
28:         $node.childNodes.Append(newNode)$;
29:     end for
30:     for $child \in node.childNodes$ do
31:         $DHC(child)$;
32:     end for
33: end function
```

remain per-target constant. The encoded target IP takes up 16B space to distinguish whether the reply is from the target address or intermediary device (e.g., “destination unreachable” replies from a firewall). Note that 6Scan only takes replies directly from the target addresses (i.e., the source address is the same as the encoded target IP) into the regional activity calculation process because our scan aims to obtain more active target addresses rather than the blocking targets behind a firewall.

Algorithm 2 Dynamic Target Generation

```
Input: A partitioned address space list $X = \{X_1, \ldots, X_n\}$; The number of budget $\mathcal{B}$;
1: $Sort(X, CmpBySize())$;   ▷ sorts the space list order by the node's size in ascending.
2: $BudgetConsumption = 0$;
3: while $BudgetConsumption \leq \mathcal{B}$ do
4:     for $d = 1$ to 32 do
5:         for $\forall X_i \in X,\ X_i.|unassingedDimension| == d$ do
6:             $T = TargetGen(X_i)$;   ▷ generates all the addresses on node $X_i$ as the targets.
7:             $X.Remove(X_i)$;
8:             $Probe(T)$;   ▷ probes the target addresses.
9:             $BudgetConsumption += T.Size()$;
10:        end for
11:        $Sort(X, CmpByActivity())$;   ▷ sorts the space list order by the node's activity in descending.
12:    end for
13: end while
```

Meanwhile, 6Scan uses the symmetric RC5 block cipher with a 32-bit block size [43] to randomize the sequence of the target address list. Other block ciphers (e.g., DES) could be used, yet the cipher strength is not critical to our application. However, unlike other scanners that scan the determinate subnet or address list, i.e., they can form the entire target addresses before probing begins, 6Scan does not determine all targets before scanning and dynamically adjusts the search directions during the scanning process. Therefore, 6Scan generates target addresses from the space tree node by node, and performs pseudo-random probing of the addresses in each node to distribute probing load and minimize the effects of rate-limiting.

V. Seed Collection

The target generation strategies require the input of the seed addresses to specify the direction of the scan, so the seeds play a critical role in the number of discovered active hosts. Previous studies have found a positive correlation between the number of seeds and the number of newly discovered active hosts [12], [14]. To this end, we explore an efficient seed address collection method in different regions to complement existing public IPv6 hitlist. In this section, we introduce our novel heuristic seeds collection method, which utilizes the announced prefixes to discover seeds at the regional level (e.g., country or AS).

Announced BGP prefixes represent the current global routing addresses, which is a subset of the entire IPv6 address space. To get more seed addresses in one region, 6Scan also implements a heuristic algorithm for active search under the announced prefixes in one region. Meanwhile, it combines an efficient de-aliasing algorithm in the process of regional active search to obtain more valuable seeds.

The following gives the detailed steps of regional-level seed collection. 6Scan first uses the RIPEstat data API [41] to form the prefix list which consists of prefixes associated with the country or AS (**Step 1**). Then, it sends 16 packets to the addresses with interface ID (IID)::1 within the subprefixes of each announced prefix (**Step 2**). For each packet, it enforces traversal of a subprefix with a different nybble. For example, given the announced prefix 2001:4800::/32, we generate 1 pseudo-random address for each 4-bit subprefix with IID ::1, namely 2001:4800:[0-f]000::1. The reason why addresses with IID ::1 are selected to probe is that previous work [44] shows that addresses with small IID often occupy the largest proportion in the collected addresses. In other words, they are more likely to be active. 6Scan uses a counter (i.e., prefix-counter) on the receiving thread to count the number of active addresses under the prefix. Prefix-counter consists of buckets which are dynamically allocated, and there are two fields in each bucket: one to store the prefix (i.e., K_i field with initial value none) and the other for counting (i.e., S_i field with initial value 0). When a reply arrives, 6Scan extracts its prefix and hashes it into the prefix counter. It stores the prefix in K_i and increases the value in S_i by one where i indicates the index of the prefix in prefix-counter (**Step 3**).

After all the arrived replies are hashed into the prefix counter, 6Scan traverses every bucket to check the value in S_i (**Step 4**). If S_i equals 16, then K_i is regarded as a candidate alias prefix. This is because the target under each subprefix within the prefix has a response, which is consistent with the alias prefix detection condition of the APD [23] algorithm. The main difference is that APD generates a random address under each subprefix, while 6Scan generates the address with IID ::1 under each subprefix. However, an address with IID ::1 is more likely to be active than a randomly generated address [44]. To this end, we add a qualification filter to determine the alias prefix. We randomly generate an address under the candidate alias prefix. Only when this address is reachable (i.e., S_i equals 17), the candidate alias prefix is considered as an alias prefix. Then, we delete the alias prefix from the prefix list to avoid further probing of the alias addresses (**Step 5**). If S_i less than 16, 6Scan obtains the subprefixes with active address and adds them to the prefix list so that subsequent iterations can generate targets under these active and non-alias prefixes.

Finally, 6Scan clears the prefix-counter, increases the length of prefix by 4 (i.e., one nybble) (**Step 6**) and goes to Step 2 for the next iteration. The iteration terminates when the prefix length exceeds 112. Using this heuristic search strategy and de-alias algorithm, 6Scan ensures that (1) probes are distributed evenly over more specific subprefixes to allow more balanced addresses sampling in one region; (2) the de-alias algorithm inspired by APD [23] effectively avoids the waste of probing resources caused by excessive scanning of the alias prefixes; and (3) the asynchronous design of 6Scan and the use of prefix-counter greatly improve the efficiency of regional-level seed collection and alias resolution.

VI. Performance Evaluation

In this section, we first evaluate the applicability of our dynamic search strategy when using various probe types. Then we compare the performance of 6Scan with representative existing alternatives in terms of hit rate and scanning efficiency using published hitlist. We also classify and analyze the addresses discovered by each search strategy. Finally, we evaluate our regional-level seed collection method and run country-level scanning to show the effectiveness of the collected seeds using our heuristic method.

All experiments were carried out in a real IPv6 network environment with 200Mbps bandwidth. We ensure good Internet citizenship as suggested by Partridge and Allman [45]. We did Internet-wide scanning on a Linux platform with an Intel Xeon Platinum 8369 CPU (3.3GHz) and 16 GB memory.

A. 6Scan Applicability

We first pre-scan Gasser's (de-aliased) hitlist [40] published on May 29th, 2021 using various probes (i.e., ICMPv6, UDP, and TCP) to form our seedsets. The hitlist is collected by active measurement with various protocols and multiple passive techniques with approximately 19.27M addresses and covering 13734 ASes. Table I lists the number of addresses (i.e., distinct replies from the hitlist scan) and the number of involved ASes of each seedset using CAIDA's IP to AS dataset [46]. We can see that our seedsets are relatively large and widely distributed. For simplicity, the name of the seedset also indicates the probe type we used in the subsequent scan experiments, e.g., we use the ICMPv6 probe when using seedset G_{ICMPv6}.

Table I also shows the number and proportion of different types of seeds, i.e., replies with or without a 6Scan payload and from the target or an intermediate device (e.g., firewall). We distinguish whether a reply is from the target or not by checking whether the source address in the response is consistent with the encoded target IP, i.e., if the source address is the same as the encoded target IP, the response is from the target; otherwise, it's a non-target reply. Note that the scan on some subnets will be blocked from an intermediate device (e.g., firewall) resulting in duplicate non-target replies, here we only consider the source address of these (non-target) replies as an active address. From the replies of the hitlist scan, we can see that the non-target replies account for a large proportion (i.e., 31.89% $\sim$ 58.96%) indicating that large numbers of passively collected addresses in the hitlist are blocked. More than one-third of TCP SYN responsive addresses, most are SYN/ACK replies, do not carry a 6Scan payload.

Next, we evaluate the support of each type of probe (i.e., ICMPv6, UDP, and TCP) for the dynamic search strategy. More specifically, we evaluate the proportion of replies carrying the original 6Scan payload and coming from the target address (i.e., the source address is the same as the encoded target IP from the 6Scan payload), because only these replies participate in the search direction adjustment process. Even though some replies without payload might come from the targets, they can't participate in search direction adjustments because they don't carry the encoded regional identifier. While some non-target replies carry a 6Scan payload, they also have no contribution to the dynamic search process. In other words, 6Scan gives more rewards to replies coming from the target with a 6Scan payload aiming to influence the subsequent scan to get more direct responses rather than blocked addresses.

Fig. 5 shows the proportion of replies that are valid (i.e., from the target with a payload) for 6Scan using different probes with various budgets. The ICMPv6 probe has the

TABLE I
CHARACTERISTICS OF THE SEEDSETS

Seedset	Probe type	Seed number	Involved ASes	Reply with payload Total = Target IP + non-Target IP	Reply without payload
G_{ICMPv6}	ICMPv6	3.67M	14314	3.67M(100%) = 2.50M(68.12%) + 1.17M(31.89%)	0(0%)
$G_{UDP/53}$	UDP/53	2.12M	11848	2.12M(100%) = 1.04M(49.15%) + 1.08M(50.85%)	0(0%)
$G_{TCP\_SYN/80}$	TCP_SYN/80	0.91M	9894	0.57M(61.74%) = 0.16M(16.94%) + 0.41M(44.80%)	0.35M(38.26%)
$G_{TCP\_ACK/80}$	TCP_ACK/80	0.40M	3507	0.40M(99.98%) = 0.16M(41.02%) + 0.24M(58.96%)	106(0.02%)

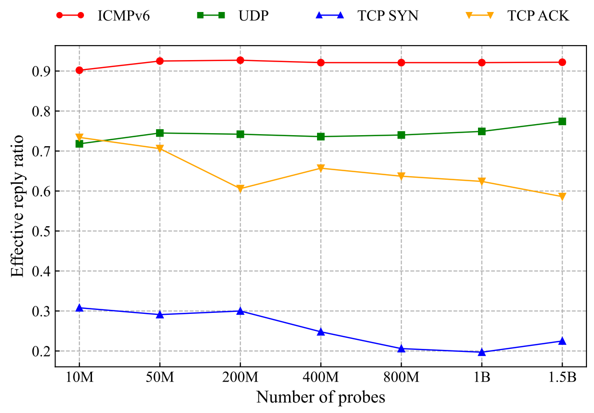

Fig. 5. Proportion of replies that are valid for 6Scan using different probes with various budgets.

maximal participation (over 90.2% replies) in the regional activity count. The UDP and TCP ACK probe, maintaining above 71.8% and 58.6% effective replies in various budgets respectively, also support 6Scan well because they also use the ICMP reply mechanism receiving a large proportion of ICMP unreachable replies. More specifically, if sending a UDP or TCP ACK probe, an ICMP unreachable error comes back (from the target) such that the target is considered active with a closed port. However, the TCP SYN probe has the lowest support for 6Scan. We found that more than half of the replies using the TCP SYN probe are from intermediate devices (i.e., the source address is different from the target), indicating that the scan is filtered. We speculate that many network administrators blocked the TCP SYN scans for security concerns.

B. Hit-Rate Performance

We compare the performance of 6Scan and other state-of-the-art target generation methods, including 6Gen [12], 6Tree [14], and 6Hit [13]. We integrate these search strategies into our probing framework and enhance these strategies to make them more suitable for large-scale network scanning with large number of seeds.

Our probing framework can get more active addresses from the replies, because our framework not only gets the replies directly from targets but also receives responses (e.g., "destination unreachable" and "time exceeded" reply) from some intermediate devices (e.g., firewall) on the path. We extract the source addresses (duplicates removed) from all types of replies as the active addresses. In fact, these non-target replies also provide a large number of discovered addresses, e.g., over 25.8% and 26.3% of newly-discovered addresses were extracted from non-target replies when running 6Scan using TCP SYN and TCP ACK probes with various budgets, respectively. Besides, we make more specific improvements for each search strategy:

- **6Gen+** We reduce the complexity of the space partition algorithm to make it scaleable to a large number of seeds. We modify the feature that seeds do not actually belong to any clusters in the hierarchical clustering process (as per Section III-A), so that a single seed can only be classified into one cluster closest to it. This reduces the time complexity of space partition from $O(n^3)$ to $O(nlogn)$.
- **6Tree+** We optimize the DHC algorithm it uses so that it can determine the number of splitting layers according to the budget. This reduces the times of downward splits, thereby accelerating the speed of space partition and improving the scanning efficiency.
- **6Hit+** The original algorithm of 6Hit may generate a large number of repeated target addresses in high-density regions due to the effect of the reinforcement learning algorithm. Distinguishing these duplicated addresses causes significant computational overhead. To this end, we optimize the target generation algorithm by utilizing symmetric RC5 block cipher [43] as used by 6Scan to conduct pseudo-random probing order so that it can generate unique addresses in each region. Meanwhile, we utilize the state encoding technique to accelerate the regional activity calculation process to improve the scanning efficiency of 6Hit.

Marginal benefit. The term "marginal benefit" in our case refers to the incremental discovered addresses on the consumption of additional probing budgets. Fig. 6 shows the hit rate and marginal benefit of 6Scan and the enhanced alternatives using the four seedsets at a maximum probing rate of 100 Kpps, the limit used by other measurement studies [3], [20]. The hit rate indicates the proportion of de-aliased *newly* discovered active addresses in the total budget.

We can see that the DHC algorithm (used by 6Scan, 6Hit+, and 6Tree+) has an obvious advantage in hit rate using a small budget (i.e., $<$ 50M) because its generated scan subspaces can wrap the seed addresses as much as possible (as discussed in Section III-A) such that to use the seeds more efficiently. However, this advantage diminishes when the budget increases, as the subspaces generated by the AHC gradually cover all seeds. Meanwhile, 6Scan shows certain advantages in seedsets G_{ICMPv6}, $G_{UDP/53}$, and $G_{TCP\_ACK/80}$ in a medium-scale budget (i.e., 50M $\sim$ 600M), indicating that its adjustment of search direction improves the hit rate performance. We also noticed that with the increase in budget, the marginal benefit and hit rate decrease sharply (i.e.,

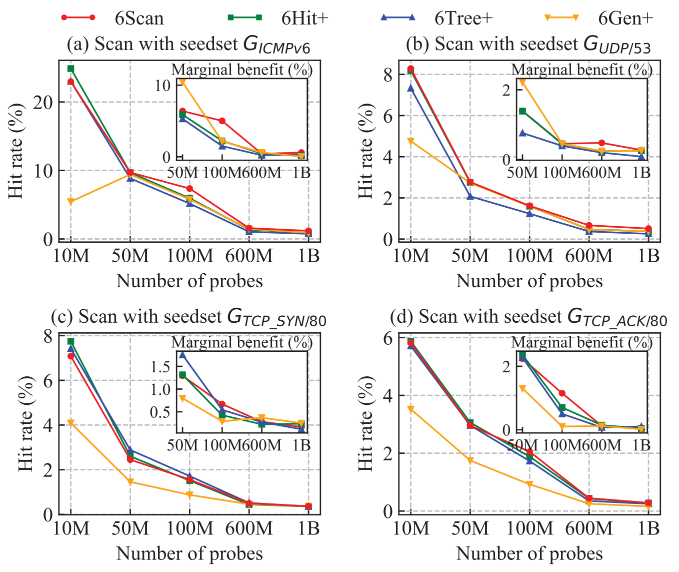

Fig. 6. Hit rate and marginal benefits of each search strategy using various probing budgets.

the marginal benefit drops to $<$ 0.6 newly-found addresses per 100 probes when increasing the budget from 600M to 1B for all the strategies on the four seedsets), indicating that the ability of seeds to guide the scanning directions becomes weaker as the scan area grows. The dynamic search direction fine-tuning on the primary scanning directions determined by the seeds can only mitigate the sharp decline in hit rate, as Fig. 6 shows that 6Scan has a relatively high marginal benefit when the budget is greater than 100M.

Billion-scale scan. We run billion-scale (i.e., 1.5B) scans to compare the total number of newly discovered addresses of each strategy. The reason we choose this scale budget is that the marginal benefit of each strategy drops to $<$ 0.03 hit per 100 probes from 1 billion to 1.5 billion budget. In other words, the utilization of seeds is almost complete on this scale budget, and it's no longer efficient to discover more active addresses. The scanning results on each seedset are shown in Table II. Note that the discovery number refers to the number of de-aliased newly discovered active addresses. Specifically, all replies exclude the addresses listed in the seedset, as well as networks and addresses in the publicly curated list of aliases from Gasser et al. [23].

Although alias addresses are widespread in the IPv6 networks, our scanning results only included a very small percentage of alias addresses because the seeds that we used excluded addresses in the alias prefixes. From the view of discovery number and hit rate, the DHC and AHC space partition algorithms can both achieve good performance. This is consistent with our discussion (as per Section III-A) that there is no significant advantage or disadvantage between the two space partition algorithms under the condition of large probing budgets. As to the strategies using the DHC algorithm (i.e., 6Scan, 6Hit+, and 6Tree+), the hit rate performance of dynamic search strategies (i.e., 6Scan and 6Hit+) is obviously better than that of static strategies (i.e., 6Tree+) demonstrating that the dynamic search direction adjustment is effective. Of note, 6Hit+'s scan results cover more ASes than 6Scan and 6Tree+, because 6Hit+ uses a more aggressive strategy to explore the large subspaces (i.e., upper-layer nodes) generated by DHC while it also brings the consumption of probing budget that affects the hit rate performance which is the typical exploration-exploitation dilemma.

Of note, the AHC algorithm (i.e., 6Gen) shows better performance in hit rate than we find in any of the DHC algorithms using seedset $G_{TCP\_SYN/80}$ and $G_{TCP\_ACK/80}$. This is because with the continuous increase of the budget, the utilization of seeds changes for both AHC and DHC algorithms, especially for the AHC algorithm which will be greatly improved in seed utilization. In other words, when the budget is small (e.g., $<$ 100M), the scan subspaces generated by DHC can cover more seeds, so DHC has higher seed utilization than AHC. With the increase in the budget, the scan subspaces generated by AHC will gradually cover all seeds too, such that its performance will gradually approach DHC and even surpass it. Recall that the higher the seed density is, the greater the probability that the subspace contains other active addresses in density-driving target generation algorithms. Similarly, if the subspaces generated by AHC (resp. DHC) to be scanned contains more seeds, the higher the seed density in the scanning address space i.e., the higher the seed utilization, such that the AHC (resp. DHC) algorithm will get better performance in hit rate.

C. Scanning Efficiency

In this section, we first define some evaluation metrics related to scanning efficiency. The *time cost* $\mathcal{T}$ refers to the total time spent when reaching the budget limit, including the time cost of the preparation before probing (i.e., address space partition) and the total probing time. The *scanning efficiency* $\mathcal{E}$ refers to the average number of new active addresses found by the scanner in a unit time, i.e., $\mathcal{E} = \frac{\mathcal{N}}{\mathcal{T}}$.

Limited scan. The *probing speed* $\mathcal{S}$ in our limited scan test refers to the average probing rate throughout the whole scanning process, namely the ratio of total budget to the total time cost, i.e., $\mathcal{S} = \frac{\mathcal{B}}{\mathcal{T}}$. The difference between it and the maximum probing rate we set can reflect the influence of a search strategy on the probing rate. Obviously, the more complex a search algorithm is the more probing speed decreases compared to the fixed maximum probing rate. We conducted a maximum probing rate of 100 Kpps in our billion-scale scans, which is also the probing rate used in the other measurement studies, e.g., [3] and [20].

We first turn to the time cost metric in Table II, the modified AHC algorithm (time complexity reduced from the original $O(n^3)$ to $O(nlogn)$), utilized by 6Gen+, is as efficient or even better than the DHC algorithm in terms of time cost in space partition. We also note that one of the dynamic scanning strategies, 6Hit+, has the worst performance in actual probing rate (the maximum probing rate is set to 100Kpps for all strategies). This is because its use of a reinforcement learning algorithm requires multiple rounds of "probing targets - calculating rewards - direction adjustment" iterations, which greatly improves the computational complexity of the scanning process. However, the actual scanning rate of 6Scan, which also uses a dynamic scanning strategy, is comparable to that of static scanning strategies (i.e., 6Tree+ and 6Gen+), because 6Scan can obtain the high-density regions in real-time by

TABLE II
SCANNING RESULTS ON EACH SEEDSET WITH 1.5 BILLION BUDGET AT MAXIMUM PROBING RATE OF 100 KPPS

Seedset	Search strategy	Alias number	Discovery number	Involved ASes	Hit rate	Time cost Total (Preparation + Probing)	Probing speed (Kpps)	Scanning efficiency (Number per Sec.)
G_{ICMPv6}	6Scan	0.14M	10.14M	9163	**0.68%**	4:32:45 (0:0:20 + 4:32:25)	91.66	**619.62**
	6Hit+	0.13M	9.74M	9414	0.65%	5:32:42 (0:0:20 + 5:32:22)	75.14	487.93
	6Tree+	0.12M	9.00M	9125	0.60%	4:24:57 (0:0:20 + 4:24:37)	94.36	566.14
	6Gen+	0.13M	9.22M	5326	0.61%	4:30:50 (0:0:18 + 4:30:32)	92.31	567.38
$G_{UDP/53}$	6Scan	0.15M	5.39M	7209	**0.36%**	4:33:30 (0:0:10 + 4:33:20)	91.40	**328.43**
	6Hit+	0.10M	4.01M	7381	0.27%	3:35:10 (0:0:10 + 5:35:00)	74.58	199.40
	6Tree+	0.14M	3.91M	7190	0.26%	4:25:53 (0:0:11 + 4:25:42)	94.02	245.09
	6Gen+	0.09M	4.38M	7747	0.30%	4:29:03 (0:0:06 + 4:28:57)	92.91	271.31
$G_{TCP\_SYN/80}$	6Scan	0.18M	4.32M	4930	0.29%	4:33:13 (0:0:04 + 4:33:09)	91.50	263.53
	6Hit+	0.17M	4.21M	4995	0.28%	5:27:43 (0:0:04 + 5:27:39)	76.28	209.53
	6Tree+	0.15M	4.18M	4826	0.27%	4:34:14 (0:0:04 + 4:34:10)	91.16	254.04
	6Gen+	0.14M	4.79M	6221	**0.31%**	4:31:28 (0:0:03 + 4:31:25)	92.09	**294.08**
$G_{TCP\_ACK/80}$	6Scan	0.16M	2.79M	2965	0.19%	4:34:07 (0:0:02 + 4:34:05)	91.20	169.63
	6Hit+	0.17M	2.69M	3001	0.18%	5:35:54 (0:0:02 + 5:35:52)	74.43	133.47
	6Tree+	0.17M	2.65M	2895	0.18%	4:34:28 (0:0:02 + 4:34:36)	91.09	160.92
	6Gen+	0.18M	2.97M	3267	**0.20%**	4:31:37 (0:0:02 + 4:31:35)	92.04	**182.24**

using the state encoding technique, and the region with the highest density will be scanned directly in the subsequent process instead of taking a large number of iterations for trade-offs in multiple high-density regions as 6Hit+ does. In terms of scanning efficiency (i.e., the average newly discovered addresses found per second), 6Scan and 6Gen+ show the best performance on different datasets. 6Scan achieves better performance on scanning efficiency using seedsets G_{ICMPv6} and $G_{UDP/53}$, which is a 9.21% and 21.05% improvement over the state-of-the-art methods, respectively.

Non-throttled scan. We measure the potential probing speed of 6Scan in our testbed (i.e., a 4-core Intel Xeon Platinum 8369 CPU @ 3.3 GHz with 16 GB memory, and a 200 Mbps network connectivity) where this capacity can be easily met by normal servers or even laptops. However, a full-speed scan will trigger traffic limiting protection on our local gateway as it could be mistaken for a DDoS attack. To this end, we negotiated with the network administrator and received permission to run a non-throttled scan within 5 minutes. Thus we run 6Scan at full speed with a budget of 50M using seedset G_{ICMPv6} to assess its potential speed. On this condition, 6Scan achieved a 163.66 Kpps probing speed with a $\approx 34.67\%$ CPU utilization and $\approx 12.67\%$ memory usage.

D. Address Analysis

We combine the scanning results discovered by each search strategy using the four seedsets. Table III shows the total discovery number. It shows clearly that 6Scan can find the largest number of newly discovered addresses (i.e., 16.78M in total), which is 0.95M more than the second largest number of newly-found addresses using 6Gen. Besides, 6Scan and 6Gen took nearly the same scanning time as shown in Table II (i.e., about 18 hours in total). In other words, compared with the state-of-the-art solutions, 6Scan can discover 6% more active addresses with nearly the same scanning time.

Major ASes. Table IV shows the top 10 ASes and the ratio of addresses within the newly-discovered addresses found by each search strategy. We can see that all strategies found the most active addresses on ASN 12322. We use the APD [23] algorithm to detect the prefixes of discovered addresses from ASN 12322, and no alias prefix is found. We also note that the major ASes discovered by each strategy are similar, indicating both DHC and AHC can generate promising scan subspaces if the budget is adequate, whereas dynamic strategies are just fine-tuning and optimizing the search direction on this basis.

Addressing patterns. An IPv6 address consists of a global routing prefix, a local subnet identifier, and a 64-bit interface identifier (IID) [47]. While the global routing prefix is determined to route traffic destined to a Local Area Network (LAN), the configuration of IID is more flexible to ensure the uniqueness of the host interface in the local network. According to RFC 7707 [48], these IIDs could be categorized into the following patterns:

EUI64-embedded where the interface's hardware 48-bit IEEE Media Access Control (MAC) address is encoded into the IID. The IID of the EUI-64 address has four specific nybbles (i.e., "ff:fe") inserted between the Organizationally Unique Identifier (OUI) and the rest of the Ethernet address. For instance, 2001:888::0250:56ff:fe89:49be.

IPv4-embedded where the address embeds a normal IPv4 address into the lowest-32 or lowest-64 bit. For instance, 2001:888::194:109:20:106.

Low-byte where all the bytes of the IID are set to zero except the least significant bytes in one or two lowest order. For instance, 2001:888::1.

Pattern-byte where more than two bytes of the IID have zero values. These IIDs contain a specific addressing pattern different from the above.

Randomized where the IID keeps a pseudo-random representation as these addresses might be privacy addresses by using Stateless Address Autoconfiguration (SLAAC) [49].

Though other patterns exist and can be identified (e.g, Port-embedded, wordy-embedded, and etc.), they are quite rare in practice [44]. Utilizing the modified addr6 tool [50], we have classified the IID pattern of the newly-discovered addresses into five categories i.e., EUI64-embedded, IPv4-embedded, low-byte, byte-pattern, and randomized. Specifically, addr6 first determines whether an address is a EUI-64 type by

TABLE III
ADDRESSING PATTERNS OF THE ACTIVE ADDRESSES DISCOVERED BY EACH SEARCH STRATEGY

Search strategy	Total discovery number	Addressing patterns				
		EUI64-embedded	IPv4-embedded	Low-byte	Byte-pattern	Randomized
6Scan	**16.78M**	1.23% (206.39K)	1.31% (219.14K)	63.77% (10.70M)	2.22% (373.73K)	31.59% (5.30M)
6Hit+	15.33M	1.20% (183.96K)	1.29% (199.56K)	62.29% (9.55M)	2.09% (321.37K)	33.13% (5.08M)
6Tree+	14.63M	0.93% (136.06K)	1.26% (185.52K)	69.92% (10.23M)	2.06% (301.52K)	27.88% (4.08M)
6Gen+	15.83M	0.80% (126.64K)	1.18% (187.69K)	67.28% (10.65M)	2.19% (346.64K)	30.83% (4.88M)

TABLE IV
TOP 10 ASES AND THE RATIO OF ADDRESSES WITHIN THE NEWLY-DISCOVERED ADDRESSES FOUND BY EACH SEARCH STRATEGY

AS name	ASN	Ratio
Free SAS	12322	26.52%
Amazon02	16509	5.34%
Chinanet	4134	3.65%
China Telecom	4812	3.51%
KPN B.V.	1136	3.50%
Amazon-AES	14618	3.05%
Man-da.de GmbH	8365	2.73%
Google-Fiber	16591	2.52%
DigitalOcean	14061	2.11%
Deutsche Glasfaser	60294	1.46%

(a) The top 10 ASes and its ratio in 6Scan's scan results.

AS name	ASN	Ratio
Free SAS	12322	32.77%
KPN B.V.	1136	8.58%
China Telecom	4812	7.32%
Sky UK Limited	5607	6.23%
Chinanet	4134	4.34%
Amazon-02	16509	2.87%
Amazon-AES	14618	2.10%
DigitalOcean	14061	1.57%
Deutsche Glasfaser	60294	1.20%
Yahoo-BF1	26101	0.97%

(b) The top 10 ASes and its ratio in 6Hit+'s scan results.

AS name	ASN	Ratio
Free SAS	12322	28.01%
Amazon-02	16509	7.21%
China Telecom	4812	5.66%
Chinanet	4134	5.25%
KPN B.V.	1136	4.56%
Sky UK Limited	5607	3.25%
Google-Fiber	16591	2.68%
Akamai	20940	2.17%
Total Play	17072	1.89%
Uninet S.A. de C.V.	8151	1.70%

(c) The top 10 ASes and its ratio in 6Tree+'s scan results.

AS name	ASN	Ratio
Free SAS	12322	22.60%
Amazon-02	16509	9.63%
China Telecom	4812	9.09%
KPN B.V.	1136	5.72%
Chinanet	4134	5.26%
Amazon-AES	14618	3.30%
Google-Fiber	16591	2.21%
Sky UK Limited	5607	1.90%
Akamai	20940	1.87%
DigitalOcean	14061	1.72%

(d) The top 10 ASes and its ratio in 6Gen+'s scan results.

checking whether it has the specific nybbles (i.e., 'ff:fe') following the OUI. Then it checks whether it is a low-byte pattern in which it defines the IID (in nybble mode) beginning with more than 9 consecutive zeros as the low-byte pattern. Next, addr6 extracts the lowest-32 and lowest-64 bits of the IPv6 address to see whether it can be decoded to a legitimate IPv4 address. Note that compared with the original addr6, we use a stricter criterion for the IPv4-embedded type, i.e., only when the decoded IPv4 address is active and it belongs to the same AS as the IPv6 address, that the IPv6 address can be identified as an IPv4-embedded type. Addr6 defines the IID with more than two zero-byte as the pattern-byte type. If it is not the above four categories, addr6 simply put them into the randomized class.

The addressing pattern classification results are shown in Table III. The low-byte pattern occupies the largest proportion of the newly discovered addresses found by each search strategy. Our heuristic seed collection approach is just based on this observation, that iteratively collects the first address under the BGP prefixes. The randomized pattern takes the second place of the most heavily represented address, which is usually generated by SLAAC for devices in practice [51]. The EUI64-embedded addresses are also widely used as an auto-configuration mechanism to generate a unique global IPv6 address. Though the EUI-64 pattern can help manage the network, this address pattern easily leaks the hardware MAC address of a device to the higher levels of the network stack [6], [52], [53] and may pose a threat to the privacy and security of the device.

TABLE V
CHARACTERISTICS OF THE SEEDSETS WHICH ARE COLLECTED BY OUR HEURISTIC SEARCH STRATEGY

Country	Seedset	Seeds number	Budget Consumption
India	H_{ICMPv6_IN}	11.83K	2.12M
America	H_{ICMPv6_US}	82.04K	10.00M
China	H_{ICMPv6_CN}	2.47K	1.08M
Brazil	H_{ICMPv6_BR}	31.41K	8.50M
Japan	H_{ICMPv6_JP}	262.97K	10.00M
Germany	H_{ICMPv6_DE}	10.72K	6.81M
Mexico	H_{ICMPv6_MX}	0.66K	0.22M
United Kingdom	H_{ICMPv6_GB}	674.80K	10.00M
Vietnam	H_{ICMPv6_VN}	58.33K	6.74M
France	H_{ICMPv6_FR}	3.52K	0.85M

E. Seed Collection

6Scan uses two ways for seed collection: one uses the hitlist published by Gasser et al. [34] and the other utilizes

TABLE VI
COMPARISON BETWEEN OUR SEEDSETS WITH GASSER'S HITLIST IN /32 PREFIX NUMBER AT COUNTRY-LEVEL

Country	Seedset	Seed number	/32 prefix	Newly discovered /32 prefix
India	H_{ICMPv6_IN}	11.83K	192	66
	G_{ICMPv6_IN}	12.69K	197	
America	H_{ICMPv6_US}	82.04K	1424	232
	G_{ICMPv6_US}	797.09K	2404	
China	H_{ICMPv6_CN}	2.47K	100	25
	G_{ICMPv6_CN}	123.76K	197	
Brazil	H_{ICMPv6_BR}	31.41K	3131	1293
	G_{ICMPv6_BR}	35.09K	2449	
Japan	H_{ICMPv6_JP}	262.97K	135	8
	G_{ICMPv6_JP}	100.97K	392	
Germany	H_{ICMPv6_DE}	10.72K	682	105
	G_{ICMPv6_DE}	575.68K	1254	
Mexico	H_{ICMPv6_MX}	0.66K	52	18
	G_{ICMPv6_MX}	3.17K	89	
United Kingdom	H_{ICMPv6_GB}	674.80K	378	72
	G_{ICMPv6_GB}	69.68K	617	
Vietnam	H_{ICMPv6_VN}	58.33K	69	26
	G_{ICMPv6_VN}	3.62K	47	
France	H_{ICMPv6_FR}	3.52K	236	34
	G_{ICMPv6_FR}	196.46K	506	

TABLE VII
DETECTED ALIAS PREFIXES IN EACH COUNTRY USING OUR HEURISTIC SEARCH STRATEGY

Country	Total number	Listed by Gasser	Newly discovered
India	11	11	0
America	482	310	172
China	4	1	3
Brazil	3	0	3
Japan	52	32	20
Germany	3	1	2
Mexico	1	0	1
United Kingdom	39	9	30
Vietnam	5	0	5
France	39	3	36

the heuristic search strategy (as per Section V) to collect the seeds within the regions/countries. In this work, we selected the top 10 countries (i.e., India, America, China, Brazil, Japan, Germany, Mexico, United Kingdom, Vietnam, and France) with the largest number of IPv6 users according to the IPv6 country deployment report [54] for country-level seed collection. Table V shows the characteristics of the seedsets collected by the heuristic search strategy using ICMPv6 probe. Note that our heuristic seed collection sets two termination conditions. One is that the length of the announced prefixes and the added prefixes with response addresses iterates to 112 (as per Section V), and the other is that the probing budget reaches 10M. For example, our simple heuristic search strategy only spent a budget of 1.08M to reach the end condition of the maximum prefix length (i.e., 112) in China, while it reached the budget limit of 10M in America and Japan.

Although our heuristic seed collection strategy uses a small probing budget in each region/country, it collected a wide range of prefixes as it searches widely across the announced BGP prefixes in the region. Table VI shows a comparison of the number of /32 prefixes contained in seedsets collected by our heuristic strategy using ICMPv6 probes with the seedset $G_{ICMPv6\_country\ code}$ (i.e., all ICMPv6 responsive addresses in the country from Gasser's hitlist published on May 29th, 2021). We can see that although the number of seeds collected in most countries (e.g., India, America, China and Brazil) is less than the number of seeds in Gasser's hitlist, the number of /32 prefixes it covers is in the same order of magnitude as that covered by Gasser's hitlist in different countries. In particular, our heuristic strategy found more /32 prefixes in Brazil and Vietnam. And the seeds we collected in different countries all found numbers of /32 prefixes that did not appear in Gasser's hitlist. We also found that Gasser's hitlist is extremely geographically unbalanced, with nearly 30% of addresses from America and less than 1% from India, Mexico, and Vietnam. Our simple and efficient seed collection method uses active measurement to find many newly discovered addresses from different countries/regions/ASes, which is a supplement of the seeds in Gasser's hitlist and alleviates its geographical imbalance problem.

In addition, we have identified a number of alias prefixes during seed collection in various countries. We verified these identified alias prefixes, and all of them meet the alias prefix determination conditions of APD [23]. Table VII shows the alias prefixes we detected in each country. Some of these prefixes have been included in Gasser's published alias prefix list and some are newly discovered. In particular, we found 172 alias prefixes not listed in Gasser's alias list with prefix length between 32 and 52 from America.

F. Country-Level Scanning

We ran 6Scan and 6Gen+ to perform country-level scanning using Gasser's hitlist and our collected seedsets with a small probing budget (i.e., 10 million) for each seedset. The reason why we select 6Scan and 6Gen+ for comparison is that they are the optimal search strategies using DHC and AHC space partition algorithms, respectively.

The scanning results in different countries are shown in Table VIII where seedset $G_{ICMPv6\_country\ code}$ and $H_{ICMPv6\_country\ code}$ indicate all the ICMPv6 responsive seed addresses from the country in Gasser's hitlist and the collected seeds from the country using our heuristic method, respectively. Using the mixed seedset (i.e., union of the regional seeds from Gasser's hitlist and seeds collected by our heuristic method), both search strategies can find more IPv6 addresses and achieve a higher hit rate. To be more specific, 6Scan achieved $1.29\% \sim 29.58\%$ hit rate while 6Gen+ achieved $0.50\% \sim 29.29\%$ hit rate for the ten country-level scans, which is respectively $0.12\% \sim 1478.45\%$ and $0.12\% \sim 405.36\%$ improvement over the hit rates obtained with Gasser's hitlist only. This demonstrates that our heuristic seed collection approach is an effective complement to existing public hitlist. The newly-found seeds can help future research relevant to regional activity discovery, and further benefit the regional security analysis in IPv6 [55].

Note that 6Scan achieved better performance than 6Gen+ in terms of hit rate on the mixed seedsets, except for the seedset from China and United Kingdom, where 6Scan and 6Gen+ have very close performance. This is consistent with our previous discussion (as per Section III-A) that DHC may be more suitable than AHC when using small or medium probing budget. We also note that both 6Scan and 6Gen+ achieved an amazing hit rate (over 8%) in the mixed seedsets from America, Japan, China, and France. We found that

TABLE VIII
COUNTRY-LEVEL SCANNING RESULTS WITH 10 MILLION BUDGET USING 6SCAN AND 6GEN+ IN DIFFERENT SEEDSET

Country	Seedset	Seed number	Discovery number		Hit rate	
			6Scan	6Gen+	6Scan	6Gen+
India	G_{ICMPv6_IN}	12.69K	139.71K	105.47K	1.40%	1.05%
	$G_{ICMPv6\_IN} \cup H_{ICMPv6\_IN}$	24.01K	155.04K	147.51K	**1.55%**	1.48%
America	G_{ICMPv6_US}	797.09K	1000.15K	880.88K	10.00%	8.81%
	$G_{ICMPv6\_US} \cup H_{ICMPv6\_US}$	876.13K	1296.34K	1057.54K	**12.96%**	10.58%
China	G_{ICMPv6_CN}	123.76K	843.23K	843.43K	8.43%	8.43%
	$G_{ICMPv6\_CN} \cup H_{ICMPv6\_CN}$	126.00K	843.63K	843.83K	8.44%	**8.44%**
Brazil	G_{ICMPv6_BR}	35.09K	321.96K	207.73K	3.22%	2.08%
	$G_{ICMPv6\_BR} \cup H_{ICMPv6\_BR}$	64.83K	343.33K	211.63K	**3.43%**	2.12%
Japan	G_{ICMPv6_JP}	100.97K	169.28K	168.47K	1.69%	1.68%
	$G_{ICMPv6\_JP} \cup H_{ICMPv6\_JP}$	363.55K	861.71K	848.89K	**8.62%**	8.49%
Germany	G_{ICMPv6_DE}	575.68K	452.55K	433.86K	4.53%	4.34%
	$G_{ICMPv6\_DE} \cup H_{ICMPv6\_DE}$	584.17K	457.39K	441.84K	**4.57%**	4.42%
Mexico	G_{ICMPv6_MX}	3.17K	614.75K	472.08K	6.15%	4.72%
	$G_{ICMPv6\_MX} \cup H_{ICMPv6\_MX}$	3.71K	627.39K	487.14K	**6.27%**	4.87%
United Kingdom	G_{ICMPv6_GB}	69.68K	379.30K	398.90K	3.79%	3.99%
	$G_{ICMPv6\_GB} \cup H_{ICMPv6\_GB}$	743.63K	550.33K	560.61K	5.50%	**5.61**%
Vietnam	G_{ICMPv6_VN}	3.62K	8.84K	12.45K	0.09%	0.12%
	$G_{ICMPv6\_VN} \cup H_{ICMPv6\_VN}$	61.88K	129.18K	49.79K	**1.29%**	0.50%
France	G_{ICMPv6_FR}	196.46K	1.94M	1.85M	19.36%	18.48%
	$G_{ICMPv6\_FR} \cup H_{ICMPv6\_FR}$	199.45K	1.96M	1.93M	**19.58**%	19.29%

these seedsets have a large number of seeds, indicating that more seed addresses can help find more promising search directions. This phenomenon implicitly suggests that the effective guidance of seeds for scanning might be restrained in the address space close to the seed vector. Therefore, to obtain a higher hit rate and scanning efficiency, we should set the probing budget a dozen times of the number of seeds yet not too much, so that the search strategy can explore the address space in an appropriate scope.

VII. CONCLUSION

In this work, we developed 6Scan, an efficient scanning tool for Internet-wide IPv6 active address search. 6Scan adopts a dynamic search strategy to adjust the subsequent search directions towards the regions of high activities. Meanwhile, 6Scan encodes the regional identifier of the target address within the packet so that the regional activity can be directly obtained from the replies. These unique features make 6Scan more efficient and effective in scanning the vast address space of IPv6. Scanning the Internet IPv6 with a large probing budget using Gasser's hitlist and a small probing budget using the mixed country-level seedsets, 6Scan has achieved much better performance on both hit rate and scanning efficiency than the state-of-the-art solutions.

ACKNOWLEDGMENT

The authors would like to greatly appreciate the anonymous reviewers for their insightful comments.

REFERENCES

[1] Z. Durumeric, E. Wustrow, J. A. Halderman, E. Wustrow, and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *Proc. USENIX Secur. Symp. (USENIX Security)*, 2013, pp. 605–620.
[2] R. D. Graham. (2022). *MASSCAN: Mass IP Port Scanner*. [Online]. Available: https://github.com/robertdavidgraham/masscan
[3] R. Beverly, "Yarrp'ing the internet: Randomized high-speed active topology discovery," in *Proc. IMC*, Nov. 2016, pp. 413–420.
[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*, Oct. 2018, pp. 308–321.
[5] R. Padmanabhan, J. P. Rula, P. Richter, S. D. Strowes, and A. Dainotti, "DynamIPs: Analyzing address assignment practices in IPv4 and IPv6," in *Proc. CoNEXT*, Nov. 2020, pp. 55–70.
[6] E. Rye, R. Beverly, and K. C. Claffy, "Follow the scent: Defeating IPv6 prefix rotation privacy," in *Proc. IMC*, Nov. 2021, pp. 739–752.
[7] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, "A look at router geolocation in public and commercial databases," in *Proc. IMC*, Nov. 2017, pp. 463–469.
[8] J. Matherly. (2022). *Shodan: The Search Engine for Internet-Connected Devices*. [Online]. Available: https://www.shodan.io
[9] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 542–553.
[10] Z. Durumeric et al., "The matter of heart-bleed," in *Proc. IMC*, 2014, pp. 475–488.
[11] P. Foremski, D. Plonka, and A. Berger, "Entropy/IP: Uncovering structure in IPv6 addresses," in *Proc. IMC*, Nov. 2016, pp. 167–181.
[12] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target generation for internet-wide IPv6 scanning," in *Proc. IMC*, Nov. 2017, pp. 242–253.
[13] 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. IEEE INFOCOM*, May 2021, pp. 1–10.
[14] Z. Liu, Y. Xiong, X. Liu, W. Xie, and P. Zhu, "6Tree: Efficient dynamic discovery of active addresses in the IPv6 address space," *Comput. Netw.*, vol. 155, pp. 31–46, May 2019.
[15] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl, "On reconnaissance with IPv6: A pattern-based scanning approach," in *Proc. IEEE Int. Conf. Availability, Rel. Secur. (ARES)*, Aug. 2015, pp. 186–192.
[16] T. Cui, G. Gou, G. Xiong, C. Liu, P. Fu, and Z. Li, "6GAN: IPv6 multi-pattern target generation via generative adversarial nets with reinforcement learning," in *Proc. IEEE INFOCOM*, May 2021, pp. 1–10.
[17] T. Yang, B. Hou, Z. Cai, K. Wu, T. Zhou, and C. Wang, "6Graph: A graph-theoretic approach to address pattern mining for internet-wide IPv6 scanning," *Comput. Netw.*, vol. 203, Feb. 2022, Art. no. 108666.
[18] 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. IEEE INFOCOM*, May 2022, pp. 1679–1688.
[19] E. C. Rye and R. Beverly, "Discovering the IPv6 network periphery," in *Proc. PAM*, 2020, pp. 3–18.
[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, "Deployment of an algorithm for large-scale topology discovery," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2210–2220, Dec. 2006.

[22] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale IPv4 alias resolution with MIDAR," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 383–399, Apr. 2013.

[23] O. Gasser et al., "Clusters in the expanse: Understanding and unbiasing IPv6 hitlists," in *Proc. IMC*, 2018, pp. 364–378.

[24] T. Albakour, O. Gasser, R. Beverly, and G. Smaragdakis, "Third time's not a charm: Exploiting SNMPv3 for router fingerprinting," in *Proc. IMC*, Nov. 2021, pp. 150–164.

[25] R. Beverly, W. Brinkmeyer, M. Luckie, and J. P. Rohrer, "IPv6 alias resolution via induced fragmentation," in *Proc. PAM*, 2013, pp. 155–165.

[26] M. Luckie, R. Beverly, W. Brinkmeyer, and K. Claffy, "Speedtrap: Internet-scale IPv6 alias resolution," in *Proc. IMC*, Oct. 2013, pp. 119–126.

[27] R. Padmanabhan, Z. Li, D. Levin, and N. Spring, "UAv6: Alias resolution in IPv6 using unused addresses," in *Proc. PAM*, 2015, pp. 136–148.

[28] K. Vermeulen et al., "Alias resolution based on ICMP rate limiting," in *Proc. PAM*, 2020, pp. 231–248.

[29] A. Marder, "APPLE: Alias pruning by path length estimation," in *Proc. PAM*, 2020, pp. 249–263.

[30] T. Fiebig, K. Borgolte, S. Hao, C. Kruegel, and G. Vigna, "Something from nothing (there): Collecting global IPv6 datasets from DNS," in *Proc. PAM*, vol. 7192, 2017, pp. 30–43.

[31] K. Borgolte, S. Hao, T. Fiebig, and G. Vigna, "Enumerating active IPv6 hosts for large-scale security scans via DNSSEC-signed reverse zones," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 770–784.

[32] A. Dhamdhere, M. Luckie, B. Huffaker, K. Claffy, A. Elmokashfi, and E. Aben, "Measuring the deployment of IPv6: Topology, routing and performance," in *Proc. IMC*, 2012, pp. 537–550.

[33] G. Song et al., "Towards the construction of global IPv6 hitlist and efficient probing of IPv6 address space," in *Proc. IWQoS*, Jun. 2020, pp. 1–10.

[34] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the IPv6 internet: Towards a comprehensive hitlist," in *Proc. Netw. Traffic Meas. Anal. (TMA)*, 2016, pp. 1–7.

[35] G. Huz, S. Bauer, K. Claffy, and R. Beverly, "Experience in using MTurk for network measurement," in *Proc. ACM SIGCOMM Workshop Crowdsourcing Crowdsharing Big (Internet) Data*, Aug. 2015.

[36] A. Yeow. (2022). *Bitnodes API*. [Online]. Available: https://bitnodes.earn.com/

[37] (2022). *RIPE NCC IPMap*. [Online]. Available: https://ftp.ripe.net/ripe/ipmap/

[38] M. Luckie, "Scamper: A scalable and extensible packet prober for active measurement of the internet," in *Proc. IMC*, 2010, pp. 239–245.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[40] (2022). *Gasser's IPv6 Hitlist Service*. [Online]. Available: https://ipv6hitlist.github.io/

[41] (2022). *RIPE NCC RIPEstat Data API*. [Online]. Available: https://stat.ripe.net/docs/data_api/

[42] G. Wan et al., "On the origin of scanning: The impact of location on internet-wide scans," in *Proc. IMC*, Oct. 2020, pp. 662–679.

[43] J. Black and P. Rogaway, "Ciphers with arbitrary finite domains," in *Proc. CT-RSA*, 2002, pp. 114–130.

[44] Q. Hu and N. Brownlee, "How interface ID allocation mechanisms are performed in IPv6," in *Proc. CoNEXT Student Workshop (CoNEXT Workshop)*, 2014, pp. 26–27.

[45] C. Partridge and M. Allman, "Ethical considerations in network measurement papers," *Commun. ACM*, vol. 59, no. 10, pp. 58–64, 2016.

[46] (2022). *CAIDA's Prefix to ASN Dataset*. [Online]. Available: https://www.caida.org/data/routing/routeviews-prefix2as/

[47] S. E. Deering and B. Hinden, *IP Version 6 Addressing Architecture*, document RFC 4291, 2006.

[48] F. Gont and T. Chown, *Network Reconnaissance in IPv6 Networks*, document RFC 7707, 2016.

[49] T. Narten, T. Jinmei, and S. Thomson, *IPv6 Stateless Address Autoconfiguration*, document RFC 4862, 2007.

[50] F. Gont. (2022). *Addr6*. [Online]. Available: https://www.si6networks.com/research/tools/ipv6toolkit

[51] X. Li, B. Liu, X. Zheng, H. Duan, Q. Li, and Y. Huang, "Fast IPv6 network periphery discovery and security implications," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 88–100.

[52] J. Martin, E. Rye, and R. Beverly, "Decomposition of MAC address structure for granular device inference," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. (ACSAC)*, Dec. 2016, pp. 78–88.

[53] S. J. Saidi, O. Gasser, and G. Smaragdakis, "One bad apple can spoil your IPv6 privacy," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 52, no. 2, pp. 10–19, Apr. 2022.

[54] G. Huston. (2022). *The IPv6 Country Deployment Reports*. [Online]. Available: http://labs.apnic.net/dists/v6dcc.html

[55] J. Czyz, M. Luckie, M. Allman, and M. Bailey, "Don't forget to lock the back door! A characterization of IPv6 network security policy," in *Proc. NDSS*, 2016, pp. 21–24.

Bingnan Hou received the bachelor's and master's degrees in network engineering from the Nanjing University of Science and Technology, China, in 2010 and 2015, respectively, and the Ph.D. degree in computer science and technology from the National University of Defense Technology, China, in 2022. His research interests include network measurement and network security.

Zhiping Cai received the bachelor's, master's, and Ph.D. degrees in computer science and technology from the National University of Defense Technology, China, in 1996, 2002, and 2005, respectively. He is currently a Full Professor with the School of Computer, National University of Defense Technology. His main research interests include network security and edge computing.

Kui Wu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Full Professor. His research interests include network science, edge computing, and network security.

Tao Yang received the B.Sc. and M.Sc. degrees in computer science and technology from the National University of Defense Technology, China, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree. His research interests include network measurement and network security.

Tongqing Zhou received the bachelor's, master's, and Ph.D. degrees in computer science and technology from the National University of Defense Technology, China, in 2012, 2014, and 2018, respectively. His main research interests include ubiquitous computing, mobile sensing, and data privacy.