Computer Networks 235 (2023) 109987

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

6Search: A reinforcement learning-based traceroute approach for efficient IPv6 topology discovery

Ning Liu, Chunbo Jia, Bingnan Hou *, Changsheng Hou, Yingwen Chen, Zhiping Cai *

College of Computer, National University of Defense Technology, Changsha, 410073, Hunan, China

ARTICLE INFO

Keywords:
IPv6
Topology discovery
Reinforcement learning algorithm

ABSTRACT

Topology discovery can infer the interconnection relationship between network entities. A complete network topology is of great significance for network security analysis, application research, etc. However, due to the huge address space and uneven distribution of active addresses in the IPv6 Internet, it is infeasible to use brute-force traceroute to discover the entire topology. To address this problem, we propose 6Search, a target generation method based on reinforcement learning algorithm for IPv6 topology discovery. 6Search first obtains the routeable BGP prefixes and then carries out traceroute in each (/32) prefix. The number of probes allocated is dynamically adjusted based on the results of previous scans. Using the reinforcement learning algorithm, 6Search allocates more probes to prefixes with more address discovery in each scan iteration. Real-world experiments demonstrate that 6Search has better performance in terms of discovery efficiency, which is 24.1%–139.8% improvement over the existing methods.

1. Introduction

With the exhaustion of IPv4 address space in 2011 [1], the next generation Internet Protocol, IPv6, with its huge address space, is beginning to accelerate its promotion and deployment worldwide. Since 1996, a series of RFC documents have been published defining and specifying IPv6. Since 2012, many well-known websites have started to support IPv6 permanently, such as Twitter, YouTube, Google, etc. In July 2022, more than 40% of Google users accessed their services via IPv6 [2].

Network topology plays a crucial role in facilitating efficient network administration. It offers valuable insights that enable administrators to optimize network configuration, identify faults, detect vulnerabilities, and monitor network communication effectively. By understanding the network topology, administrators can promptly pinpoint the location of any issues within the network, allowing for quick resolution [3–5]. It is also crucial for network neighbor discovery [6], security analysis [7], and Internet modeling [8]. However, obtaining the network topology of large-scale IPv6 networks poses significant challenges. Due to the vast address space of IPv6, mapping out the topology of a specific region or the entire Internet requires extensive traceroute measurements. Our goal is to minimize the number of probes used while still obtaining comprehensive information. This process is essential for network administrators to gain a comprehensive understanding of the network and effectively manage its operations. Therefore, we propose an efficient topology discovery method for the IPv6 Internet that discovers more active IPv6 addresses (i.e., the route hops) within limited probes.

In previous IPv4 measurement studies, the computing power of modern processors is sufficient to support Internet-wide scans in a short time. Measurement tools such as Yarrp [9] and Flashroute [10] can effectively discover the topology and complete the trace route of all /24 prefixes in the IPv4 address space within an hour. In addition, ZMap [11] can scan the entire IPv4 Internet within one hour and obtain the information of nearly 4 billion online devices, which is useful for researchers to perform address analysis [12,13] and geographical positioning [14]. However, these tools become extremely inefficient in the IPv6 Internet because the address space of IPv6 is 2^{96} (roughly equal to 10^{29}) times that of IPv4. The massive address space leads to the inefficiency of tools that use the brute-force scanning approach. Researchers have proposed efficient technologies [15–17] for IPv6 address scanning in the huge address space.

Inspired by 6Hit [18], an active IPv6 address scanning technique using a reinforcement learning algorithm, we propose 6Search, a reinforcement learning based target generation method, which generates targets in IPv6 prefixes and scans the targets to perform topology discovery. 6Search adjusts the search directions according to the previous scan results in each iteration, and considers the number of responses as

* Corresponding authors.
E-mail addresses: liuning_nudt@nudt.edu.cn (N. Liu), jiachunbo@nudt.edu.cn (C. Jia), houbingnan19@nudt.edu.cn (B. Hou), houchangsheng@nudt.edu.cn (C. Hou), ywch@nudt.edu.cn (Y. Chen), zpcai@nudt.edu.cn (Z. Cai).

https://doi.org/10.1016/j.comnet.2023.109987
Received 12 April 2023; Received in revised form 20 July 2023; Accepted 16 August 2023
Available online 19 August 2023

the reward for the next iteration to allocate more probes in the high-reward prefix. We measure the complexity of the network topology with different evaluation criteria, e.g., the number of active nodes and the number of network links, etc. We propose the concept of detection efficiency for topology discovery, defined as the ratio of the total number of active addresses discovered to the total number of probes sent. We consider it as a metric because the more active addresses or hops found in a prefix means we get a more comprehensive topology in the prefix. In other words, we try to use fewer probes or budgets to get a more comprehensive IPv6 network topology.

6Search first obtains all IPv6 routable BGP prefixes. It then allocates probes to each prefix according to the expectation ratio of each prefix to perform active topology discovery and counts the number of newly discovered addresses in each prefix. 6Search modifies the expectation of each prefix according to the number of addresses found in each iteration and adjusts the subsequent search directions. This reinforcement learning feedback mechanism directs the probes to those prefixes with more newly discovered addresses, improving scanning efficiency. In addition, 6Search uses the Doubletree algorithm [19] to reduce redundant detection.

The main contributions of our work are two-fold. First, we introduce 6Search, a reinforcement learning-based method for efficient IPv6 topology discovery, which generates targets within IPv6 prefixes and uses a limited number of probes to discover a comprehensive network topology in the vast IPv6 address space. Second, we demonstrate through real-world experiments that in our test sets 6Search outperforms other topology discovery technologies, achieving scan efficiency improvements ranging from 24.1% to 139.8%.

The rest of this paper is organized as follows. In Section 2, we introduce the traditional topology discovery methods. In Section 3, we describe the detail of 6Search. In Section 4, we evaluate the performance of 6Search in real IPv6 Internet. Finally, we conclude in Section 5.

2. Background and related work

Over the years, researchers have conducted extensive research on network topology discovery and proposed various solutions to improve its performance. In the IPv4 Internet, Traceroute is the most widely used tool for active topology discovery. It detects the number of route interfaces between the host sending the packet and the destination by sending packets with increasing TTL values to track the hops and listening for the response from the route interface. Although Traceroute plays an significant role in active network topology discovery, it also suffers from the challenges of load balancing, redundancy detection, etc. To overcome these issues, researchers have proposed a series of techniques.

The Doubletree algorithm is used for topology discovery to reduce redundant detection. The network topology, from a vantage point to the Internet, resembles a tree-like structure, allowing the selection of an intermediate node to initiate detection. The algorithm functions by commencing detection at an intermediate node, progressing to the source node, and halting when it reaches the vantage point or detects a previously discovered interface. To address false links and loops caused by routers using load balancing strategies in the measurement results, Augustine et al. proposed an improved method called Paris-Traceroute [20]. This method maintains a consistent stream ID for a series of probe packets sent to the same destination address. Veitch et al. later introduced the Multipath Detection Algorithm (MDA) [21] and MDA-Lite [22] to further improve the impact of load balancing strategy on the scan results.

Spring et al. [5] proposed Rocketfuel to directly measure ISP topology at the router level in the IPv4 Internet, using two technologies to reduce the measurement budget. The first technology is directed probing, which uses BGP routing information to select only those tracing routes that can pass through the mapped ISP. This technology

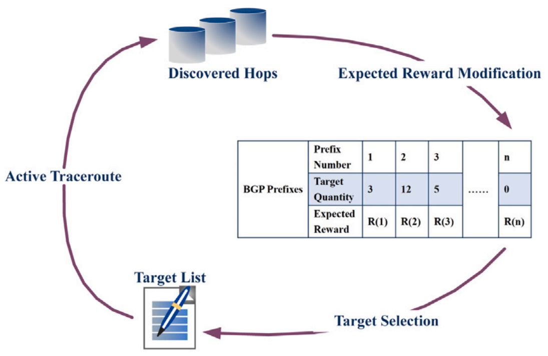


Fig. 1. The operation process of 6Search.

can use routing information to skip unnecessary tracing routes, but it can lead to false negatives when BGP is updated. The second technology is path reductions, which consists of ingress reduction, egress reduction and next-hop AS reduction. Ingress reduction is that the target only needs to be detected by one of the vantages if two vantages share an ISP entry; Egress reduction is that when two prefixes share an ISP exit, only one of them needs to be tracked; Next-hop AS reduction is that if the AS number of the next hop of the two prefixes is the same, only one of them needs to be tracked. These two technologies enable Rocketfuel to reduce the number of traces required to map an ISP by three orders of magnitude, while maintaining high accuracy. However, as the scale of the Internet has grown, the effect of path reductions has diminished and false negatives have increased. The underlying detection methods of Rocketfuel are now outdated, and new measurement tools have become more efficient [23,24]. For example, Marechal et al. proposed Anaximander which infers the ISP router level mapping without compromising the final topology coverage while reducing the detection budgets.

Traditional traceroute technology requires the maintenance of detection status for each transmission, and it can only increase the TTL value for a new round of detection after receiving feedback from the previous round. However, Yarrp [9] and Flashroute [10] have revolutionized the underlying logic of traditional traceroute technology. Yarrp, inspired by Zmap, introduced stateless scanning technology that allows it to traceroute all /24 prefixes in the IPv4 address space within an hour. By incorporating all necessary states into the probe, Yarrp eliminates the need to maintain a connection state. Yarrp also separates the sending thread from the receiving thread, which greatly improves the parallelism of scanning and speeds up the scanning process. Moreover, Yarrp follows the lead of Paris-Traceroute, which uses the same source/destination port and protocol ID in all discoveries to reduce false links and loops caused by load balancing routers. Yarrp6 [25], an updated version of Yarrp, added support for IPv6 network scanning. Although Yarrp and Yarrp6 increase the scanning speed, they do not address the problem of redundant scanning.

Similar to Yarrp, Flashroute also adopts the stateless scanning technology to separate the sending thread from the receiving thread by editing all the required information in the probe header, which is returned in the response payload. Meanwhile, Flashroute adopts a variety of technologies to optimize detection. Flashroute adopts the Doubletree algorithm to reduce redundant detection of previously discovered route interfaces. Flashroute maintains a “Destination Control Block” (DCB) to support its probing logic for each destination and to track the detection progress in both forward and backward directions. In addition, Flashroute introduces a pre-detection process before the main detection to obtain the hop distance to some destination nodes, and performs backward and forward detection based on the hop distance as the split point of Doubletree. Flashroute can complete a full /24 IPv4 scan in less than 20 min.

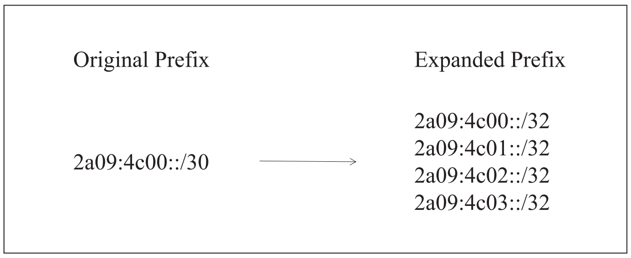


Fig. 2. A toy example of expanding the prefix to /32 granularity.

Table 1
Number and percentage of routable prefixes extracted from RIPE data API.

Prefix size	Number	Proportion
/28	151	0.25%
/29	14701	24.75%
/30	175	0.29%
/31	146	0.25%
/32	29758	50.09%
/35	117	0.20%
/36	1138	1.92%
/40	679	1.14%
/44	935	1.57%
/47	145	0.24%
/48	11016	18.54%

3. Design of 6search

6Search is used for large-scale topology discovery of the IPv6 Internet to obtain a relatively comprehensive network topology with fewer probing budgets. We introduce an overview of 6Search, and then describe its three key technical components in detail: (1) reinforcement learning-based target generation; (2) doubletree algorithm; and (3) prevention of early convergence.

3.1. Overview of 6search

Fig. 1 illustrates the operation of 6Search, which is a traceroute tool used for efficient discovery of the IPv6 address space. 6Search first retrieves IPv6 BGP prefixes using the RIPE data API, which provides a list of IPv6 routable prefixes for 231 countries/regions. If the prefix length is less than 32, 6Search expands it to /32 granularity as shown in Fig. 2. 6Search chooses to unify prefixes larger than /32 to /32 granularity because the majority of routable prefixes extracted from the RIPE data API are at /32 granularity (accounting for 50.09%), as shown in Table 1. Unifying them to the same granularity ensures that the scan exploration opportunities are evenly distributed among the /32 subnets within these larger prefixes. Furthermore, 6Search sets an expected reward for each prefix to allocate probing budgets. In each iteration, 6Search updates the expected reward based on the scan results and adjusts the new budgets for each prefix accordingly.

3.2. Reinforcement learning-based target generation

To achieve high scanning efficiency, it is critical to allocate probing resources within the address space contained by the /32 prefixes. 6Search addresses this challenge by utilizing the reinforcement learning-based target generation method, which is a feedback mechanism that allows it to concentrate resources in regions with more active nodes. This approach is effective because it optimizes the allocation of resources based on the real detection results, increasing the likelihood of detecting highly active networks.

In regions with low activity levels, the reinforcement learning approach will allocate fewer budgets because there is a lower probability of discovering active addresses. If the detection process fails to find any new nodes despite allocating some resources, 6Search will terminate the detection process in these prefixes.

Algorithm 1 Target generation based on reinforcement learning.

Input: The IPv6 prefix set P, the total detection budget B, and the number of targets per iteration b.
Output: Response address set N.

```
1:  $TotalBudget = 0$
2:  $N = \emptyset$
3:  while $TotalBudget \leq B$ do
4:      $N^* = \emptyset$
5:      $budget = 0$
6:      $\{t_1, t_2, ...t_b\} = Target\_Generation(P, B, b)$
7:      for $i = 1$ to $b$ do
8:          $N^*$.add($Topo\_Detection(t_i)$)
9:          $budget$ += $Topo\_Detection(t_i)$
10:     end for
11:     $N = N \bigcup N^*$
12:     $TotalBudget$ += $budget$
13:     $Reward\_Update(N^*)$
14: end while
15: return $N$
```

The target generation algorithm based on reinforcement learning is shown in Alg. 1. The algorithm takes the set P={P_1, P_2...P_n} of all prefixes of a country, the total detection budgets B and the number of detection targets b in each iteration as input. The granularity of the prefixes is 32. The algorithm iterates through the following three processes: (1) $Target\_Generation(P, B, b)$ which generates a different number of targets in each prefix according to the expected reward. (2) $Topo\_Detection$ which carries out scanning process. (3) $Reward\_Update$ which updates expected rewards based on the detection results. Next we introduce the three processes in detail.

Step 1: $Target\_Generation(P, B, b)$. 6Search reallocates resources after receiving feedback to achieve higher detection efficiency. 6Search based on reinforcement learning needs to fully explore the environment (i.e. the address space composed of all prefixes) and select appropriate target allocation strategies to improve its detection efficiency. Therefore, 6Search adopts the soft-max action selection algorithm to reward active regions. The allocation of actions in the Softmax algorithm is based on the Boltzmann distribution and its formula is as follows:

$$P(i) = \frac{e^{\frac{R(i)}{\tau}}}{\sum_{j=1}^{n} e^{\frac{R(j)}{\tau}}},$$

where $R(i)$ represents the expected reward value of region i; $P(i)$ represents the probability of generating detection targets in region i. Therefore, the number of targets allocated in region i for each iteration of detection follows a binomial distribution: $X \sim B(b, p(i))$ and its mathematical expectation $b * p(i)$ is the number of targets generated in region i for each iteration. Gasser et al. [26] collect active IPv6 addresses from multiple sources, and we can download them from ipv6 hitlist [27]. The active addresses in hitlist exist in only a few prefixes, while most other prefixes have no corresponding active addresses in hitlist. Therefore, we randomly generate addresses instead of downloading active IPv6 addresses from hitlist to explore broader space and more regions.

Step 2: $Topo\_Detection$. After generating targets, we send probes to these targets for topology discovery, and use Doubletree algorithm to reduce redundant detection. Then we collect the detection results which count the detection packets sent and record the newly discovered nodes in each region to update the expected reward R.

Step 3: $Reward\_Update$. As mentioned above, we update the expected reward based on the detection results of the previous iteration so that the regions with more complex network topology get more

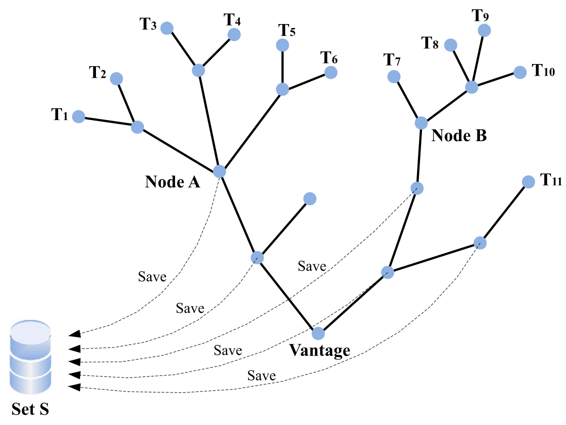


Fig. 3. Topography obtained from a single vantage.

Algorithm 2 Doubletree Algorithm

```
Input:
    Target set T
Output:
    Topography obtained by T

1: split_point = 16                                    ▷ set the split point
2: $S = \emptyset$
3: $Topo = \emptyset$
4: for $i = 0$ to $m$ do
5:     $ttl$ = split_point
6:     while $ttl > 0$ do
7:         $Middle\_node$ = traceroute($ttl$)
8:         if $Middle\_node \in S$ then
9:             break
10:        else
11:            $S$.add($Middle\_node$)
12:            $Topo$.add($Middle\_node$, $ttl$)
13:            $ttl = ttl - 1$
14:        end if
15:    end while
16:    while $ttl < 32$ do
17:        $Middle\_node$ = traceroute($ttl$)
18:        if $Middle\_node = T_i$ then
19:            break
20:        else
21:            $Topo$.add($Middle\_node$, $ttl$)
22:            $ttl = ttl + 1$
23:        end if
24:    end while
25: end for
26: return $Topo$
```

detection resources in the next iteration. In the first iteration, we distribute b detection targets evenly across all prefixes, and the value of $R(i)^1$ is b/n, where n represents the number of prefixes.

After iteration t, the expected reward $R(i)^{t+1}$ of region i is updated as follows:

$$R(i)^{t+1} = (1-\alpha) * R(i)^t + \alpha * r(i)^t,$$

where $\alpha(0 < \alpha \leq 1)$ is the learning rate and when it approaches 1, the previous detection effect is less which means that the learning rate is fast. $R(i)^t$ is the expected reward of the last iteration. $r(i)^t$ is the reward value obtained by the statistics of the returned results, and its formula is as follows:

$$r(i)^t = S(i)^t * X(i)_{granularity},$$

where $S(i)^t$ represents the number of nodes newly discovered in region i in this iteration. $X(i)_{granularity}$ represents the granularity of area i. As mentioned earlier, most prefixes have a granularity of 32, and a few other prefixes have a granularity greater than 32, so the address space of different regions is not exactly the same. So we introduce $X(i)_{granularity}$ to make the reward more fair.

We assume that the average number of probe packets sent to each target address is β. So the time complexity of target generation algorithm is O($\frac{B}{b*\beta} * n$), where n is the number of regions.

3.3. Doubletree algorithm

In the process of topology detection, we use Doubletree algorithm to reduce redundant detection. As mentioned above, the tree-structured network topology makes the redundant detection mainly occur in the nodes near the vantage. Therefore, Doubletree algorithm can overcome the redundant detection near the source node and reduce the repeated detection of some intermediate nodes near the vantage, but the formed topology is still a relatively complete network topology.

Alg. 2 shows the process of the Doubletree algorithm. It maintains a set S to store the discovered nodes. It first probes from an intermediate node (TTL=h) to the source node in a hop-by-hop descending manner. Before each iteration, it traverses the set S to find out if the node is in the set. If the node is not in the set, it is added, and the algorithm continues to detect backwards until TTL=1. However, if the node is already present in set S, the probe to the vantage point is stopped, and the algorithm starts probing to the destination node from h+1. Fig. 3 shows a topography obtained from a single vantage. The vantage maintains a set S and traverses the topology from T_1 to T_{11}. It adds node A to set S after detecting T_1. Then, when detecting T_2, T_3, T_4, T_5, T_6, the redundant detection from vantage to node A is stopped. Similarly, after node B joins the set S, it also reduces the redundant detection of T_8, T_9 and T_{10}.

3.4. Prevention of early convergence

As the iterations increase, 6Search will allocate more resources to regions with high detection efficiency and eventually converge on a few prefixes. If some prefixes outperform most other prefixes at the initial stage of the iteration, 6Search will converge on these regions prematurely. However, 6Search will fail to explore some regions with complex network topography and a large number of active nodes, because these regions cannot obtain resources from the early stage of exploration. This situation is an exploration–exploitation dilemma which can cause 6Search to fall into a local optimum and be unable to explore more regions. 6Search adopts two mechanisms to prevent early convergence: (1) prefix extraction and (2) generation of new prefix.

(1) **Prefix Extraction.** According to the Boltzmann formula in Section 3, if the expected reward value *(R(i))* of a region i is much larger than that of other prefixes, then the region i will obtain the majority of detection resources in the next iteration. As a result, 6Search will prematurely converge on a few regions that outperform other regions in the early stages. This means that in the subsequent exploration, 6Search cannot fully explore most other regions, even though they have a complex network topography and a large number of active nodes.

Therefore, to prevent convergence at the initial stage of the iteration, 6Search extracts the prefixes where the number of targets obtained is greater than the threshold at each iteration. We set the threshold to $\frac{1}{10}b$, where b is the total number of targets for each iteration. In the subsequent iterations, the extracted prefixes will obtain the resources for detection separately, and the reinforcement learning algorithm will no longer count these prefixes.

(2) **Generation of New Prefix.** We use the generation of new prefix mechanism to improve the exploration ability of 6Search. We use this mechanism when the number of generated prefixes is less than the threshold. We empirically set the threshold to $\frac{1}{20}n$, where n

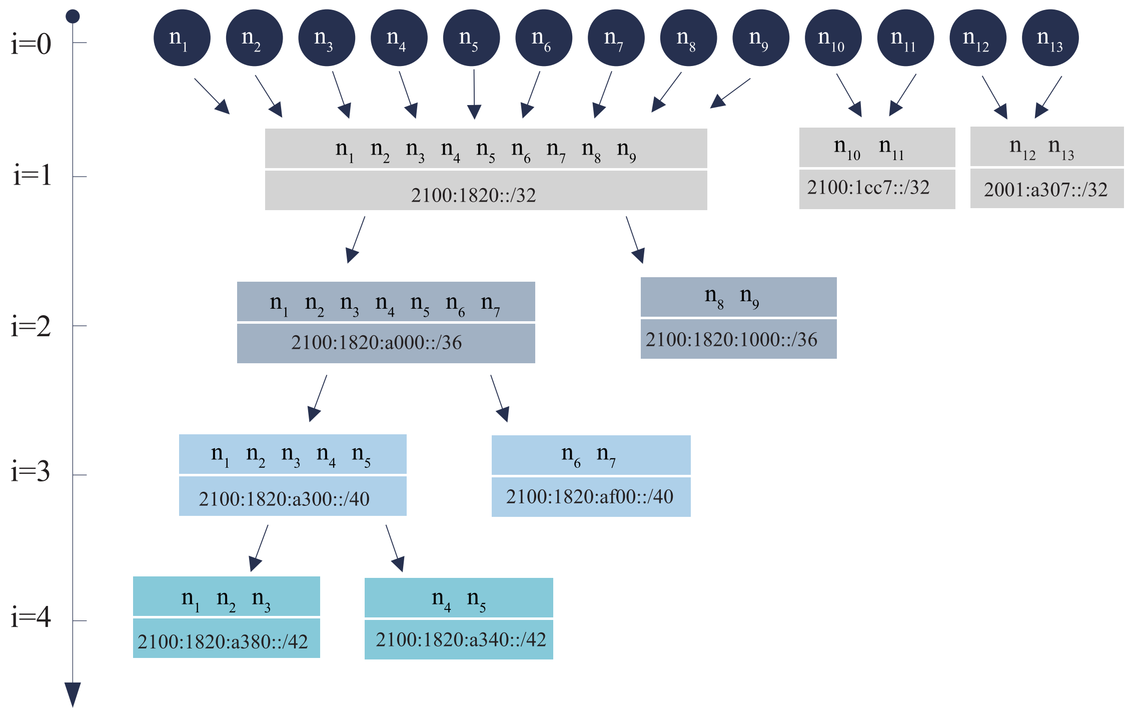


Fig. 4. An example of the generation of new prefixes.

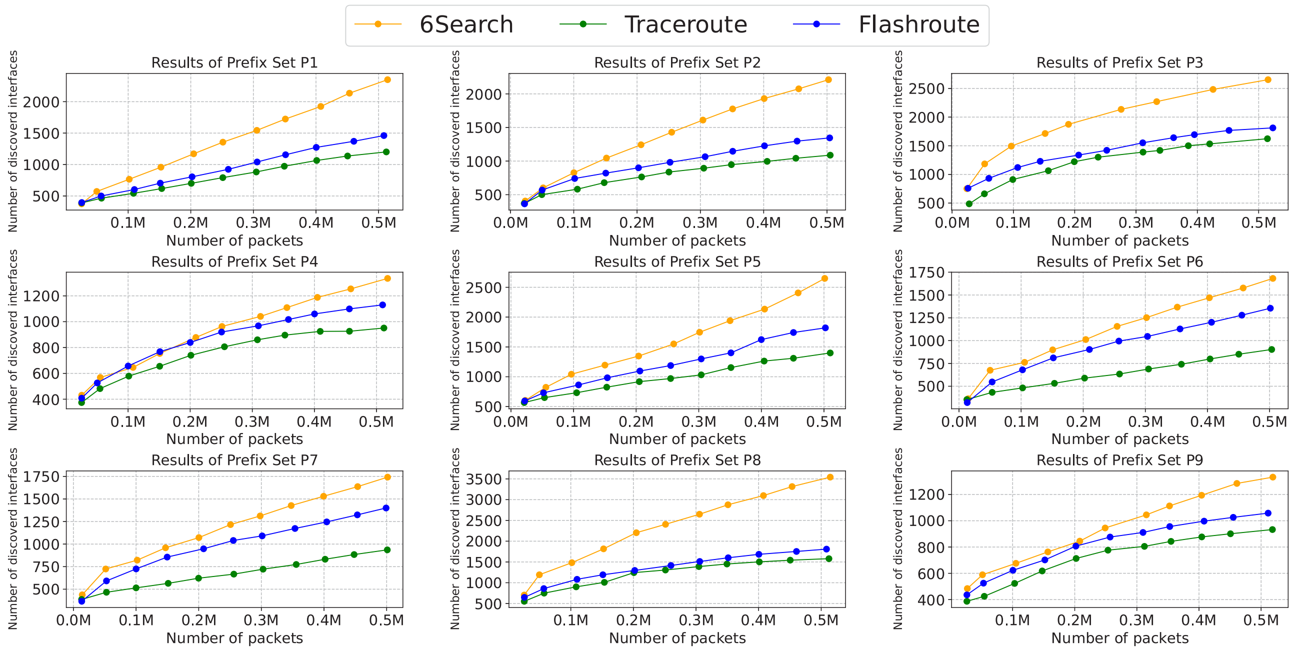


Fig. 5. Detection results in each country prefix set. 6Search can find a more comprehensive topology when using the same number of probes.

is the number of prefixes. The generation of new prefix mechanism selects the discovered active nodes to generate new prefixes and these nodes make new prefixes embed more information about the detection results. Alg. 3 shows the two processes of the prefix generation, e.g., *Node_Aggregation()* and *Prefix_Split()*.

Node_Aggregation() selects the discovered active nodes and aggregates them into different prefixes with a granularity of 32. If the number of discovered active nodes in some prefixes is less than the threshold ($\frac{n}{m}$) which is set to the average number of nodes in all /32 prefixes, these prefixes with 32 granularity can be used directly for subsequent detection. Otherwise, *Prefix_Split()* process needs to be performed.

The *Prefix_Split()* splits the prefix recursively, so that the number of discovered active nodes contained in decomposed prefixes is less than the threshold. Specifically, it adopts an algorithm (*Split()*) similar to Divisive Hierarchical Clustering (DHC) [28] to split the 32 prefixes into more granular prefixes. This process helps 6Search generate more detailed prefixes, which is conducive to the more accurate adjustment of the detection direction.

Fig. 4 shows an example of the generation of new prefixes (GP) which is employed to ensure that the scan does not overly focus on a

Algorithm 3 Generation of New Prefix

```
Input:
    Set N composed of n active nodes
Output:
    New prefix set $P$

1: $P = \emptyset$
2: $\{P_1, P_2, ...P_m\} = Node\_Aggregation(N)$
3: for $i = 1$ to $m$ do
4:     $Prefix\_Generation(P_i)$
5: end for
6: return $P$
7: function NODE_AGGREGATION(N)
8:     $P\_Set = \emptyset$
9:     for $i = 1$ to $n$ do
10:        if $Prefix\_32 \notin P\_Set$ then
11:            $P\_Set.add(Prefix\_32(N_i))$
12:        end if
13:    end for
14:    return $P\_Set$
15: end function
16: function PREFIX_GENERATION(p)
17:    if $length(p) > \frac{n}{m}$ then
18:        $\{p_1, p_2, ...p_k\} = Split(p)$
19:        for $i = 1$ to $k$ do
20:            $Prefix\_Generation(p_i)$
21:        end for
22:    else
23:        $P$.add($p$)
24:    end if
25: end function
26: function SPLIT(A)
27:    $S = \emptyset$
28:    $A^* = Prefix(A)$                          ▷ $A^*$ is the granularity of A
29:    for $i = 1$ to $t$ do
30:        $p = Granularity(a_i, A^* + 1)$            ▷ finer granularity
31:        $S(p).add(a_i)$
32:    end for
33:    return $S$
34: end function
```

limited number of prefixes, thereby enhancing its exploration capability. GP first aggregated newly discovered addresses into /32 prefixes where 3 new /32 prefixes are generated in the example. Since the number of addresses/nodes in 2100:a307::/32 exceeds the pre-defined threshold i.e., the number of discovered addresses divided by the number of generated prefixes (9 > 13/3). The prefix 2100:a307::/32 needs to be subdivided into smaller sub-prefixes through splitting until the splitting stops when the number of addresses in the newly-generated prefixes is below the threshold. In the example, the GP process ends up with six newly-generated prefixes: 2100:1cc7::/32, 2100:a307::/32, 2100:1820:1000::/36, 2100:1820:af00::/40, 2100:1820:a380::/42, and 2100:1820:a340::/42.

4. Performance evaluation

We compare the performance of 6Search with other existing topology discovery technologies, i.e., Flashroute and Traceroute, through real-world tests.

4.1. Dataset preparation

We randomly selected 9 countries from the top 15 countries with the highest deployment rate, as reported in the IPv6 country resource report by G. Huston [29]. The details of these prefixes are shown in Table 2. Since the active addresses in hitlist are contained in only a few prefixes, we randomly generate addresses as targets in each prefix so that the detection can cover all prefixes. Meanwhile we specifically select the prefixes containing active addresses, and use the active addresses in hitlist as targets to carry out topology discovery in real-world networks, and compare the detection efficiency of 6Search, Flashroute and Traceroute. For Flashroute and Traceroute, We splice the prefix part and the randomly generated non-prefix part into an IPv6 address, and then take it as the detection target.

Table 2
Prefix-related information.

Prefix set	Country	Number of prefix	Address range
P_1	China	60165	2001:250::/35–2a10:1cc7::/32
P_2	France	14801	2001:660::/32–2a12:f7c0::/32
P_3	Russia	17268	2001:640::/32–2a12:fac7::/32
P_4	Netherlands	12677	2001:504:34::/48–2a13:7::/32
P_5	Italy	10052	2001:678:12::/48–2a12:ff87::/32
P_6	Japan	10337	2001:200::/32–2a12:a307::/32
P_7	Australia	10157	2001:360::/35–2a11:8087::/32
P_8	Brazil	8774	2001:1820::/32–2a00:aee7::/32
P_9	Poland	6179	2001:678:120::/48–2a12:d007::/32

Table 3
Prefix with active address.

Prefix Set	Country	Number of prefix with active address	Total number of active address
P_1	China	906	335517
P_2	France	495	171203
P_3	Russia	578	124261
P_4	Netherlands	800	199321
P_5	Italy	238	14878
P_6	Japan	587	215645
P_7	Australia	329	48180
P_8	Brazil	3917	112072
P_9	Poland	301	35329

4.2. Real-world detection

We set the parameters empirically: the temperature $\tau = 1$; the learning rate $\alpha = 0.1$. The total budget B is 0.5M. The target number of the initial iteration is $b = 2 * n$, where n is the number of prefixes. As the experiment progresses, b decreases correspondingly to increase the number of iterations. We then compared the detection efficiency of 6Search with other topology discovery technologies in the IPv6-based Internet. Meanwhile, we strictly regulate our own behavior and comply with Internet rules [30]. We conduct the network topology discovery experiment through Alibaba Cloud server and we use AMD EPYC 7T83 and 32 GB memory for multi-thread topology discovery on the Linux platform. The detection rate is limited to 400 Kbps, and this amount of detection traffic will not cause any warning from the Internet. We use ICMPv6 probes for detection, which is the most classical method for topology discovery. We provide the source code and it is available at https://github.com/ngnnb/6Search. The experiment was performed in December 2022.

We use the IPv6 address prefix set $P_i (i \in [1, 9])$ of different countries to compare the detection efficiency of 6Search, Flashroute and Traceroute. We stop detection when the number of detection packets sent exceeds the total budget B. The detection results are shown in Fig. 5. We can see that the detection efficiency of 6Search based on reinforcement learning is significantly higher than that of Flashroute and Traceroute. The detection efficiency of Flashroute is higher than that of Traceroute, which mainly benefits from the reduction of redundant detection by the Doubletree algorithm. This shows the advantage of 6Search, which can dynamically adjust the detection direction according to the feedback

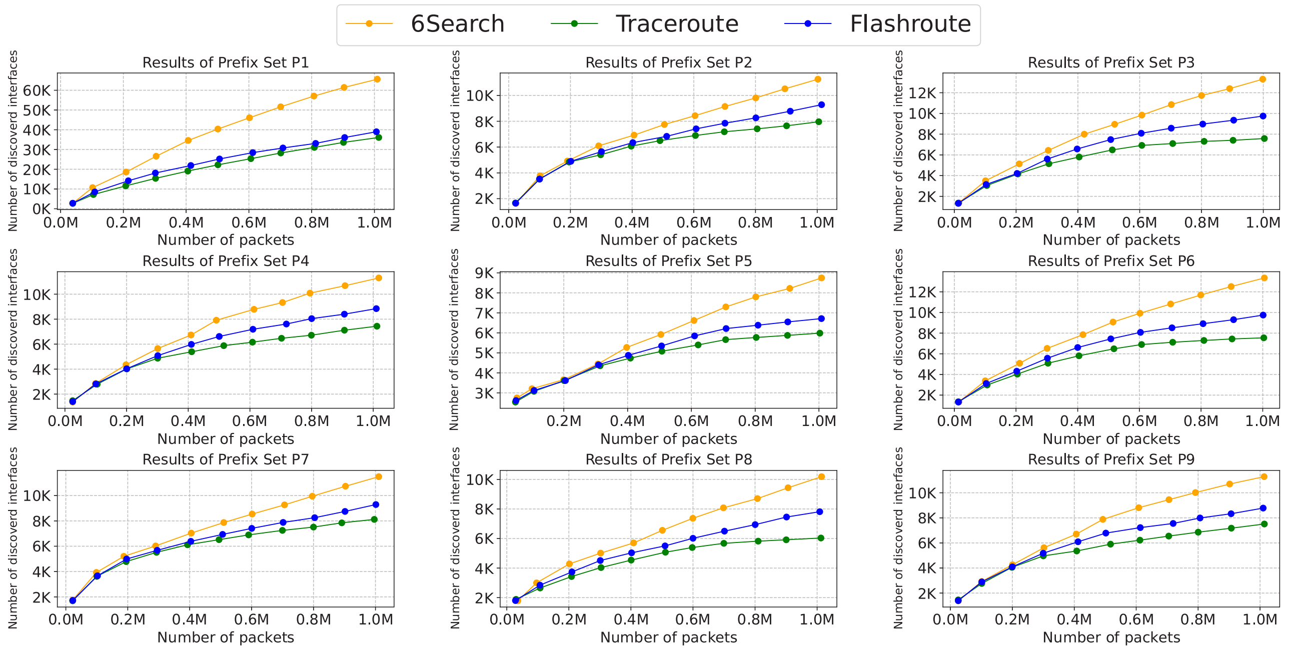


Fig. 6. Detection results with active addresses as targets.

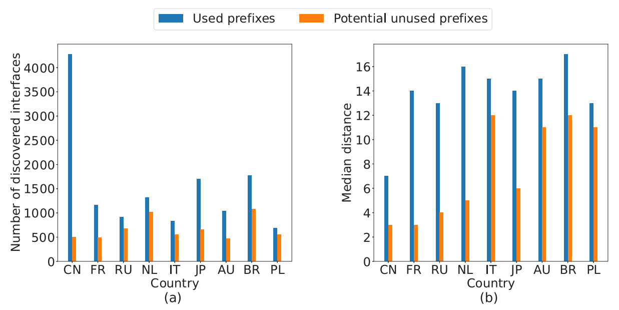


Fig. 7. The number of discovered interfaces and the median distance to the last response hop of the used and potential unused prefixes.

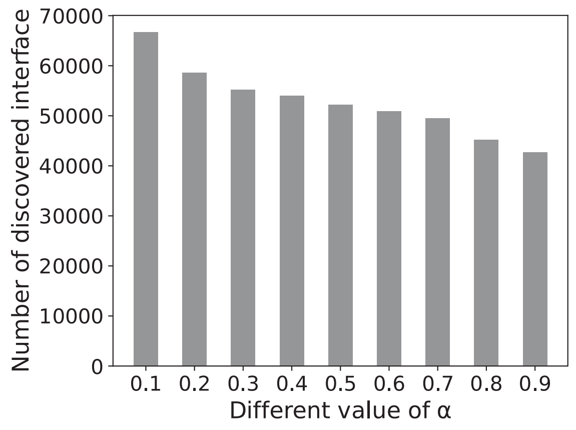


Fig. 8. Detection results of α with different values.

of each iteration, and always concentrate the detection resources in regions with more complex network topology and more active nodes. However, the detection efficiency of 6Search and other technologies is generally low (no more than 0.5%).

Therefore, we select the active IPv6 addresses from hitlist as the detection target to conduct a new real-world experiment and the total budget B is 1M. We again select all /32 prefixes of the above countries and match them with active addresses in hitlist. Table 3 shows the details of the prefixes with active addresses.

Fig. 6 shows the detection efficiency of different methods using the active addresses in hitlist as targets. The results indicate that 6Search outperforms Flashroute and Traceroute in terms of detection efficiency. In addition, we can see that prefixes that contain addresses from hitlists (i.e., used prefixes) performs significantly better than prefixes that do not contain addresses from hitlists (i.e., potential unused prefixes).

To validate the aforementioned opinion, we selected the prefixes of 9 countries and divided them into two groups. One group consists of used prefixes, while the other group consists of potential unused prefixes. For each group of prefixes, we randomly generated 10,000 targets to probe. The results of the traceroute measurements are presented in Fig. 7. Fig. 7(a) illustrates the number of discovered interfaces in each country. It is evident that the number of interfaces found when using the used prefixes is significantly higher than when using potential unused prefixes. We also measured the distances from the last response hop to our prober. The median distances for each country, when using both types of prefixes, are shown in Fig. 7(b). The median distance towards targets in used prefixes is significantly larger than in potential unused prefixes. This indicates that some routers might not have a route to those targets in the potential unused prefixes, resulting in the inability to forward the probing packets sent to the potentially unused prefixes.

4.3. Effect of parameter selection on results

We select 906 prefixes from China that contain active addresses in hitlist to test the effect of parameter selection on topology discovery results. According to the Boltzmann formula and the expected reward formula (introduced in Section 3), α and τ have similar effects, which affect the update speed of the expected reward and the convergence speed of the algorithm.

We first assess the effect of α (learning rate) on topology discovery by varying its value from 0.1 to 0.9. The total number of probe

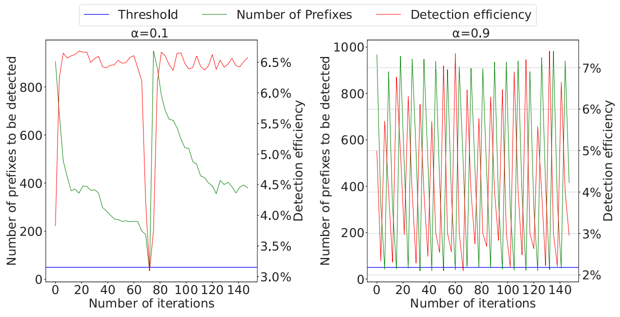


Fig. 9. The detection efficiency and prefix number of 6Search with iterations increases (α=0.1 and α=0.9).

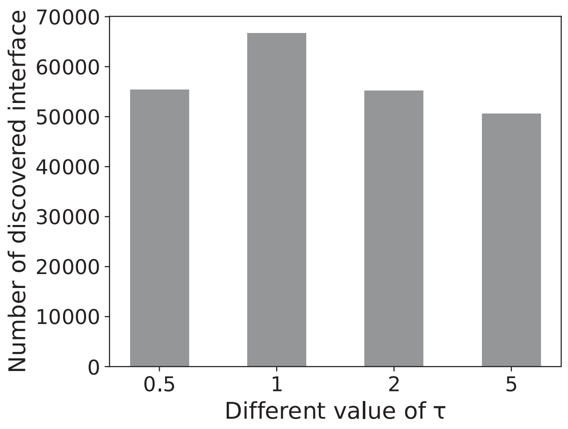


Fig. 10. Detection results of τ with different values.

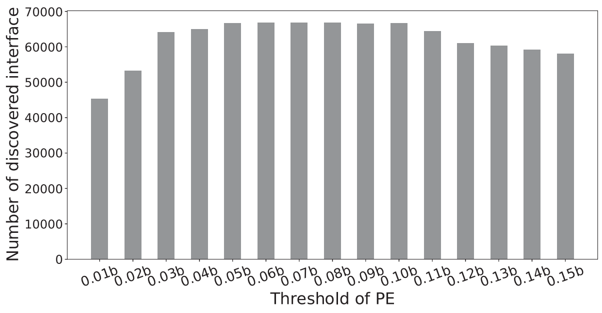


Fig. 11. Number of discovered interfaces with varying PE values.

packets for each experiment is 1,000,000. Fig. 8 shows the number of discovered interfaces for different values of α. The results indicate that the detection efficiency is the highest when α is 0.1. As the value of α increases, the number of discovered interfaces gradually decreases. Specifically, when α is 0.1, the detection efficiency is 6.56%, which is 53.7% higher than that when α is 0.9.

Fig. 9 shows the detection efficiency and the number of prefixes that can generate targets in each iteration when α takes different values (0.1 and 0.9). We can see that when α=0.1, the detection efficiency of 6Search increases to about 6.5% after the start of the iteration, while the number of prefixes that can generate targets gradually decreases. When the number of prefixes is less than the threshold, 6Search will adopt generation of new prefix mechanism to prevent convergence, which improves the ability to explore more regions. When α=0.9, 6Search quickly converges to a few active regions after starting the iteration due to its fast 'learning rate', and the detection efficiency is low. Even though 6Search has the generation of new prefix mechanism, it cannot overcome the premature convergence in the subsequent detection.

This indicates that when the value of α is small, the update speed of the expected reward is slow and 6Search can reward regions with better performance, while still having the ability to explore regions with poor performance in initial iterations. Meanwhile, the generation of new prefix mechanism allows 6Search to have stronger exploration ability and improve the detection efficiency. However, if the value of α is large, 6Search will quickly converge to the regions with high detection efficiency after a few iterations. The fast learning rate is not conducive for 6Search to explore the unknown regions, because they do not perform well in some iterations. In other words, 6Search loses its ability to explore unknown regions due to the large value of α.

Then we test the effect of the τ (temperature) and each experiment involves a total of 1,000,000 probe packets. Fig. 10 shows the number of discovered interfaces when τ takes different values. We can see that when τ is 1, the detection efficiency is the highest. As the value of τ decreases, the algorithm becomes greedier and the convergence rate is faster, which reduces its ability to explore unknown regions. On the other hand, as the value of τ increases, the reward mechanism of the reinforcement learning algorithm becomes less effective and tends to explore more regions. Therefore, when the value of τ is 1, the algorithm can achieve the balance between exploration and reward, thus achieving the highest detection efficiency.

Next, we analyze the sensitivity of parameters PE (prefix extraction) and PG (prefix generation), to prevent premature convergence and avoid getting stuck in local optima, respectively.

We selected 906 prefixes from the IPv6 hitlists and allocated a total probing budget of 1M to conduct the sensitivity analysis. We examined the impact of PE by varying its threshold from 0.01b to 0.15b, where b represents the number of probes used per iteration. Fig. 11 demonstrates that the scan efficiency is maximized when the threshold of PE is set between 0.05b and 0.1b. Based on this analysis, we chose to set the value of PE to 0.1b for the experiment.

Fig. 12 illustrates the number of discovered interfaces as the PG values range from n/15 to n/30, where n represents the initial number of prefixes used in the scan. It can be observed that when the PG value is set to a larger value (e.g., n/5), 6Search performs poorly. This could be attributed to the reinforcement learning algorithm generating new prefixes for scan exploration before gathering sufficient valuable information, resulting in a lower discovery rate. Conversely, when the PG value is set to a smaller value (e.g., n/50), 6Search also exhibits suboptimal performance. This might be due to the algorithm getting trapped in a local optimum as the iteration progresses, where the stringent threshold condition of PG fails to trigger scan exploration for the newly generated prefixes.

In conclusion, our measurements are derived from several experiments conducted on the real IPv6 Internet, leading to a statistical conclusion. In our test sets, 6Search consistently outperforms alternative methods, exhibiting detection efficiency improvements ranging from 24.1% to 139.8%. Especially when active addresses are selected as targets and the detection efficiency of all methods is greatly improved, 6Search still has a better experimental effect. This is because 6Search has a feedback mechanism based on reinforcement learning that allocates more resources to regions with more complex topography and more active nodes. Meanwhile, the generation of new prefix mechanism gives 6Search more opportunities to change the detection direction and improve the exploration ability.

4.4. Limitations

One current limitation of our study is that we rely on a single origin/prober located in Hangzhou, China for the measurement study.

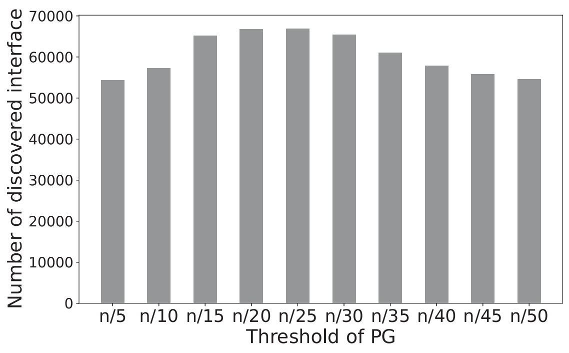


Fig. 12. Number of discovered interfaces with varying PG values.

This inevitably introduces measurement biases due to factors such as permanent and temporary blocking, packet loss, geographic biases, transient outages [31], and Internet censorship [32]. Although we have taken steps to minimize these biases by exclusively using ICMPv6 probes, which are considered less invasive than UDP and TCP probes, it is not possible to completely eliminate these biases. In future work, we intend to incorporate probes deployed in different regions to calibrate and refine our measurement results.

5. Conclusion

In this paper, we proposed 6Search, a target generation method based on reinforcement learning which is used for efficient topology discovery in the IPv6 Internet. 6Search learns the topology information in the address space contained by each prefix according to the results of each iteration, and dynamically adjusts the detection direction to detect active regions more deeply. The results in real-world tests show that 6Search achieves a higher detection efficiency compared to Flashroute and Traceroute.

CRediT authorship contribution statement

Ning Liu: Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing. **Chunbo Jia:** Writing – original draft, Writing – review & editing. **Bingnan Hou:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Changsheng Hou:** Writing – original draft, Project administration, Supervision. **Yingwen Chen:** Writing – review & editing, Project administration, Supervision. **Zhiping Cai:** Supervision, Resources, Funding acquisition.

Declaration of competing interest

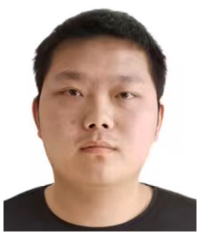
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my code in the manuscript.

References

[1] I. Livadariu, K. Benson, A. Elmokashfi, A. Dhamdhere, A. Dainotti, Inferring carrier-grade NAT deployment in the wild, in: IEEE INFOCOM, 2018, pp. 2249–2257.
[2] Google, IPv6 adoption statistics, 2022, URL https://www.google.com/intl/en/ipv6/statistics.html.
[3] Y. Jin, S. Renganathan, G. Ananthanarayanan, J. Jiang, V.N. Padmanabhan, M. Schroder, M. Calder, A. Krishnamurthy, Zooming in on wide-area latencies to a global cloud provider, in: ACM SIGCOMM, 2019, pp. 104–116.
[4] Y. Shavitt, E. Shir, DIMES: Let the internet measure itself, ACM SIGCOMM Comput. Commun. Rev. 35 (5) (2005) 71–74.
[5] N. Spring, R. Mahajan, D. Wetherall, Measuring ISP topologies with rocketfuel, ACM SIGCOMM Comput. Commun. Rev. 32 (4) (2002) 133–145.
[6] A.S.A.M.S. Ahmed, R. Hassan, N.E. Othman, IPv6 neighbor discovery protocol specifications, threats and countermeasures: a survey, IEEE access 5 (2017) 18187–18210.
[7] K. Borgolte, S. Hao, T. Fiebig, G. Vigna, Enumerating active IPv6 hosts for large-scale security scans via DNSSEC-signed reverse zones, in: S&P, IEEE, 2018, pp. 770–784.
[8] R. Pastor-Satorras, A. Vespignani, Evolution and Structure of the Internet: A Statistical Physics Approach, Cambridge University Press, 2004.
[9] R. Beverly, Yarrp'ing the internet: Randomized high-speed active topology discovery, in: ACM/USENIX IMC, 2016, pp. 413–420.
[10] Y. Huang, M. Rabinovich, R. Al-Dalky, Flashroute: Efficient traceroute on a massive scale, in: ACM/USENIX IMC, 2020, pp. 443–455.
[11] Z. Durumeric, E. Wustrow, J.A. Halderman, Zmap: Fast internet-wide scanning and its security applications, in: USENIX Security Symposium, Vol. 8, 2013, pp. 47–53.
[12] D. Plonka, A.W. Berger, kIP: a measured approach to IPv6 address anonymization. CoRR abs/1707.03900 (2017), 2017, arXiv preprint arXiv:1707.03900.
[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: Network and Distributed Systems Security (NDSS), 2016.
[14] Q. Scheitle, O. Gasser, P. Sattler, G. Carle, HLOC: Hints-based geolocation leveraging multiple measurement frameworks, in: IEEE TMA, 2017, pp. 1–9.
[15] B. Hou, Z. Cai, K. Wu, T. Yang, T. Zhou, 6Scan: A high-efficiency dynamic internet-wide IPv6 Scanner With Regional encoding, IEEE/ACM Trans. Netw. (2023).
[16] T. Yang, Z. Cai, B. Hou, T. Zhou, 6Forest: an ensemble learning-based approach to target generation for internet-wide IPv6 scanning, in: IEEE INFOCOM, 2022, pp. 1679–1688.
[17] T. Yang, B. Hou, Z. Cai, K. Wu, T. Zhou, C. Wang, 6Graph: A graph-theoretic approach to address pattern mining for internet-wide IPv6 scanning, Comput. Netw. 203 (2022) 108666.
[18] B. Hou, Z. Cai, K. Wu, J. Su, Y. Xiong, 6Hit: a reinforcement learning-based approach to target generation for internet-wide IPv6 scanning, in: IEEE INFOCOM, 2021.
[19] B. Donnet, P. Raoult, T. Friedman, M. Crovella, Efficient algorithms for large-scale topology discovery, in: ACM SIGMETRICS, 2005, pp. 327–338.
[20] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, R. Teixeira, Avoiding traceroute anomalies with Paris traceroute, in: ACM/USENIX IMC, 2006, pp. 153–158.
[21] B. Augustin, T. Friedman, R. Teixeira, Multipath tracing with Paris traceroute, in: IEEE Workshop on End-To-End Monitoring Techniques and Services, 2007, pp. 1–8.
[22] K. Vermeulen, S.D. Strowes, O. Fourmaux, T. Friedman, Multilevel MDA-lite Paris traceroute, in: ACM/USENIX IMC, 2018, pp. 29–42.
[23] E. Marechal, P. Mérindol, B. Donnet, ISP probing reduction with anaximander, in: PAM, Springer, 2022, pp. 441–469.
[24] J.-R. Luttringer, Y. Vanaubel, P. Mérindol, J.-J. Pansiot, B. Donnet, Let there be light: Revealing hidden MPLS tunnels with TNT, IEEE Trans. Netw. Serv. Manag. 17 (2) (2019) 1239–1253.
[25] R. Beverly, R. Durairajan, D. Plonka, J.P. Rohrer, In the IP of the beholder: Strategies for active IPv6 topology discovery, in: ACM/USENIX IMC, 2018, pp. 308–321.
[26] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S.D. Strowes, L. Hendriks, G. Carle, Clusters in the expanse: Understanding and unbiasing IPv6 hitlists, in: ACM/USENIX IMC, 2018, pp. 364–378.
[27] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S.D. Strowes, L. Hendriks, G. Carle, Ipv6 hitlist service, 2022, URL https://ipv6hitlist.github.io/.
[28] G. Shobha, S. Rangaswamy, Machine learning, Handbook of Statist. 38 (2018) 197–228.
[29] G. Huston, IPv6 BGP table reports, 2022, URL https://bgp.potaroo.net/index-v6.html.
[30] C. Partridge, M. Allman, Ethical considerations in network measurement papers, Commun. ACM 59 (10) (2016) 58–64.
[31] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow, Z. Durumeric, On the origin of scanning: The impact of location on internet-wide scans, in: ACM/USENIX IMC, 2020, pp. 662–679.
[32] J. Zirngibl, L. Steger, P. Sattler, O. Gasser, G. Carle, Rusty clusters? Dusting an IPv6 research foundation, in: ACM/USENIX IMC, 2022.



Ning Liu received the bachelor's degrees in Network Engineering from National University of Defense Technology, China, in 2021. He is currently pursuing the master's degree with the College of Computer, National University of Defense Technology. His research interests include network measurement and network security.

Bingnan Hou received the bachelor's and master's degrees in Network Engineering from Nanjing University of Science and Technology, China, in 2010 and 2015, respectively. He received Ph.D. degree in College of Computer, National University of Defense Technology, China, in 2022. He is currently a postdoc in College of Computer, National University of Defense Technology. His research interests include network measurement and network security.

Changsheng Hou received the bachelor's degree in School of Information and Communication Engineering from University of Electronic Science and Technology of China, in 2019. He received master's degree in College of Computer, National University of Defense Technology, China, in 2021. He is currently a Ph.D. student in College of Computer, National University of Defense Technology. 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 National University of Defense Technology, China, in 1996, 2002, and 2005, respectively. He is currently a full professor in College of Computer, National University of Defense Technology. His main research interests include network security and edge computing.