AddrMiner: A Comprehensive Global Active IPv6 Address Discovery System

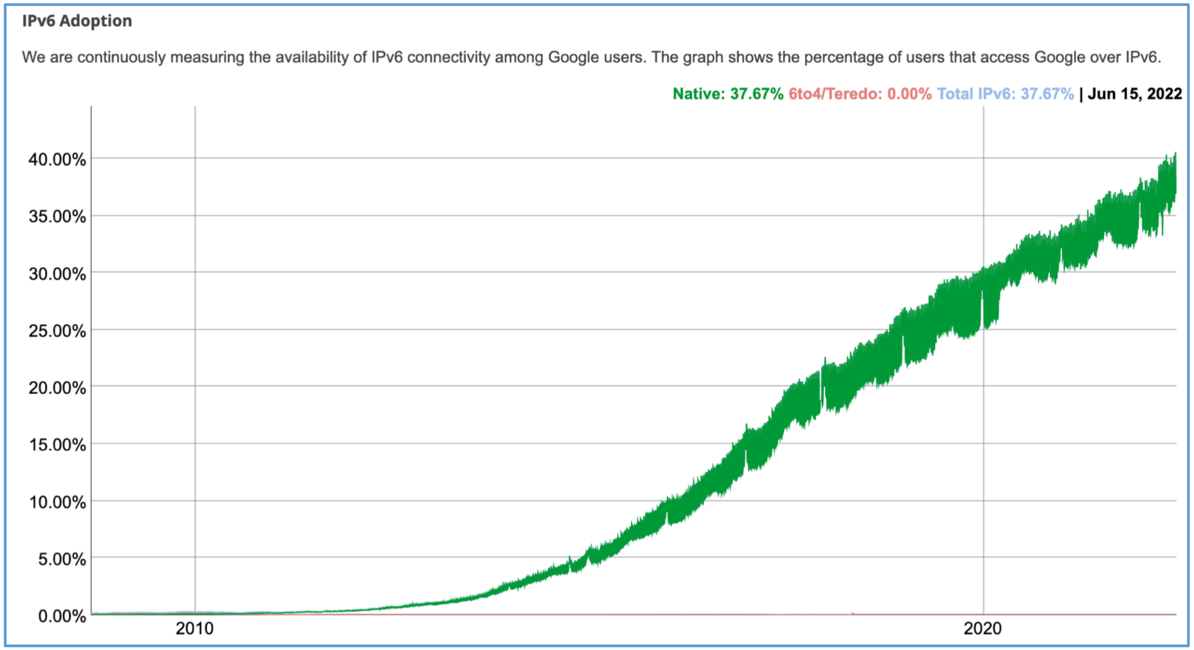
Guanglei Song, Jiahai Yang, Lin He, Zhiliang Wang, Guo Li,

Chenxin Duan, Yaozhong Liu, Zhongxiang Sun

Institute for Network Sciences and Cyberspace, Tsinghua University
Quan Cheng Laboratory, Jinan, Shandong, China

■ Background

With the growing address exhaustion of IPv4 , IPv6 is being deployed increasingly commonly around the world, and this trend will accelerate.



IPv6 Adoption

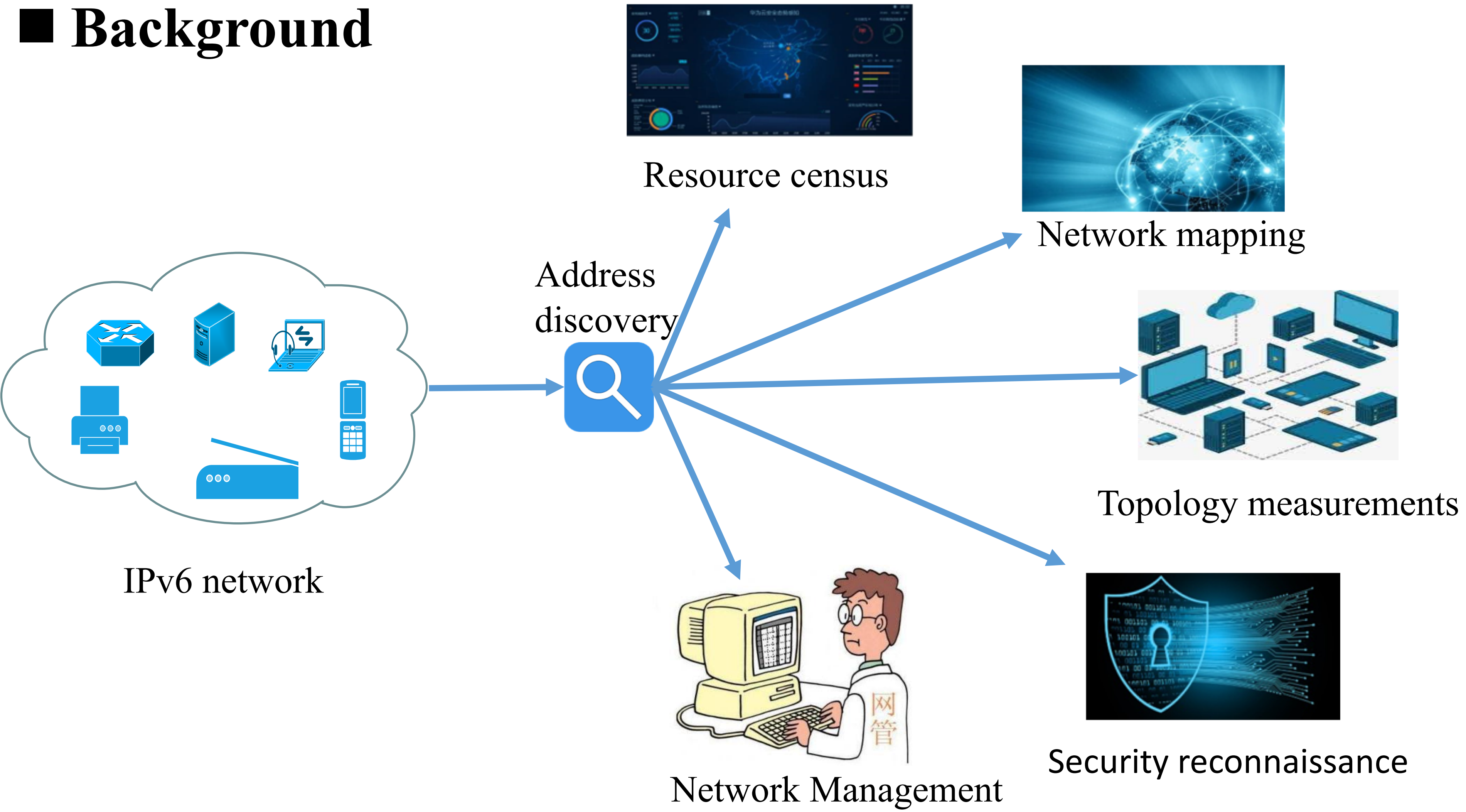
We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

Native: 37.67% 6to4/Teredo: 0.00% Total IPv6: 37.67% | Jun 15, 2022

IPv6 Adoption

Source: https://www.google.com/intl/en/ipv6/statistics.html

■ Background



IPv6 network

Address discovery

Resource census

Network mapping

Topology measurements

Security reconnaissance

Network Management

■ Motivations

- Various IPv6 address configuration methods
- Vast IPv6 space
- Low address usage

Scanning entire IPv4 address space only needs tens of minutes 😊

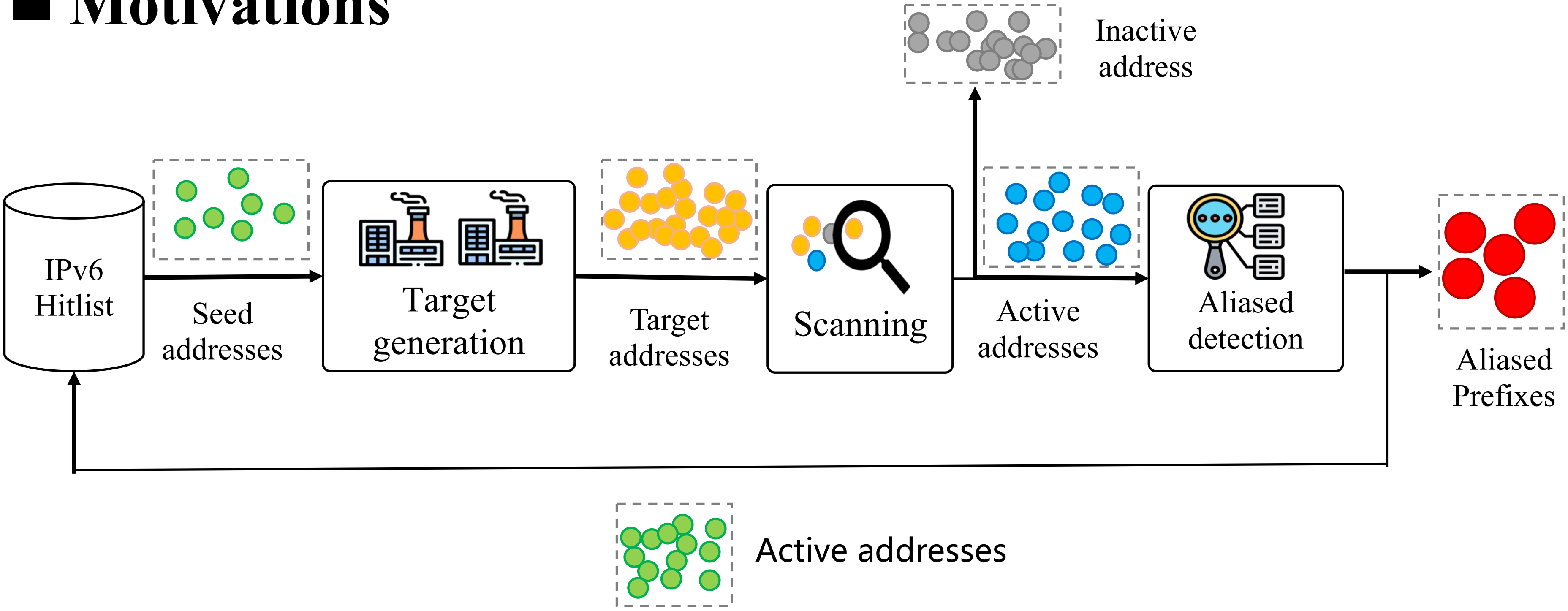
Scanning entire IPv6 address space needs more than **1 million years** 😲

↓

Brute-force scanning of all IPv6 space is infeasible

How to quickly find active IPv6 addresses in limited probe resources?

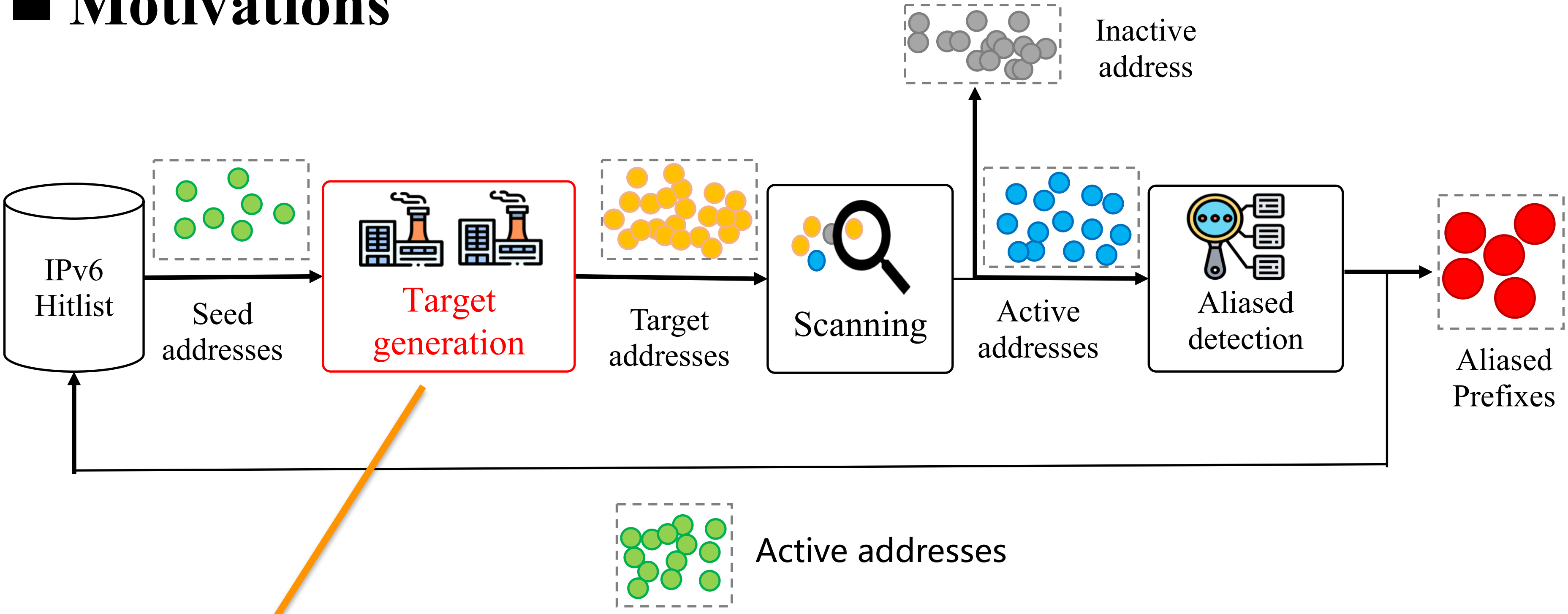
■ Motivations



Framework of IPv6 address probing

Hitlist : IPv6 address list extracted from multiple data sources
Seed addresses: Active address as input of address generation algorithms
Target address: Possible active address generated by a address generation algorithm

■ Motivations

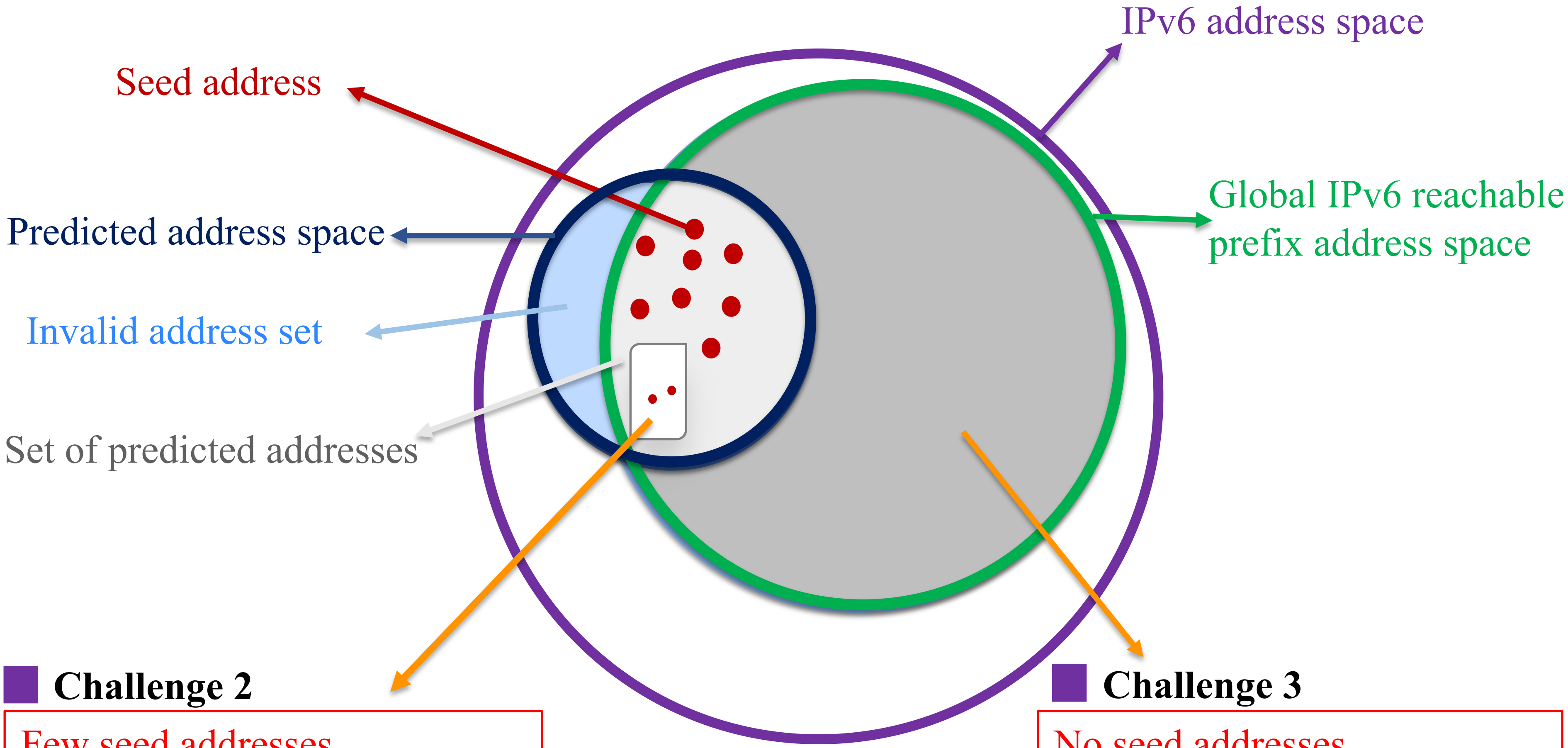


Framework of IPv6 address probing

Challenge 1

Over-dependence on seeds and poor results due to sampling bias of seeds

■ Motivations



Challenge 2

Few seed addresses,
difficult to learn characteristics

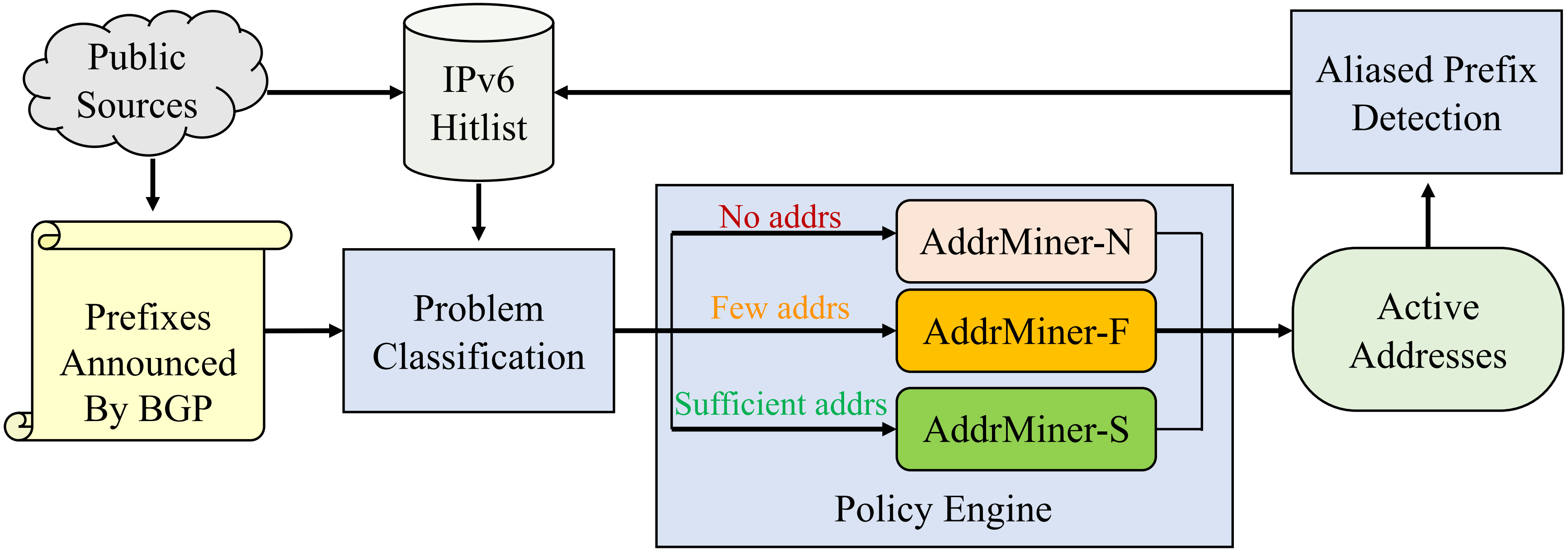
Challenge 3

No seed addresses,
unable to learn characteristics

■ Motivations

How to detect active IPv6 addresses in a comprehensive, systematic and efficient way?

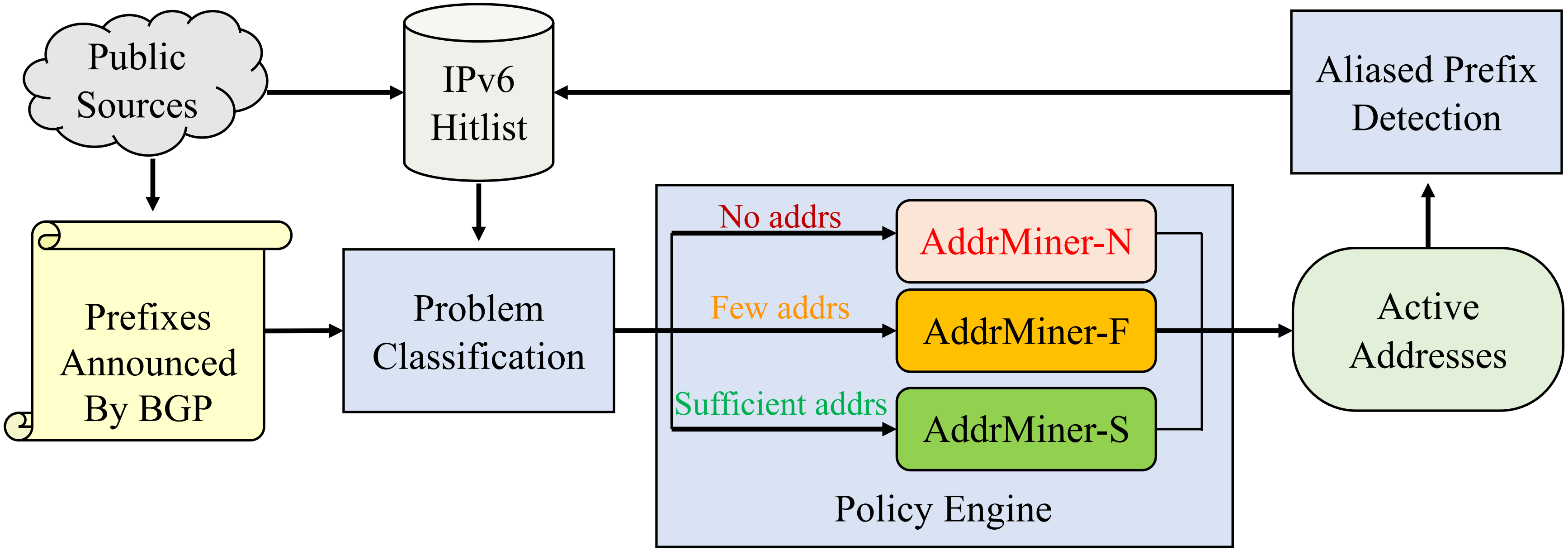
■ AddrMiner *A Comprehensive Global Active IPv6 Address Discovery System*



High-level overview of AddrMiner

AddrMiner implementation: https://github.com/AddrMiner/AddrMiner

■ AddrMiner



High-level overview of AddrMiner

AddrMiner implementation: https://github.com/AddrMiner/AddrMiner

AddrMiner-N

Address patterns (i.e., structure) tend to have similarities across network configurations

E.g.

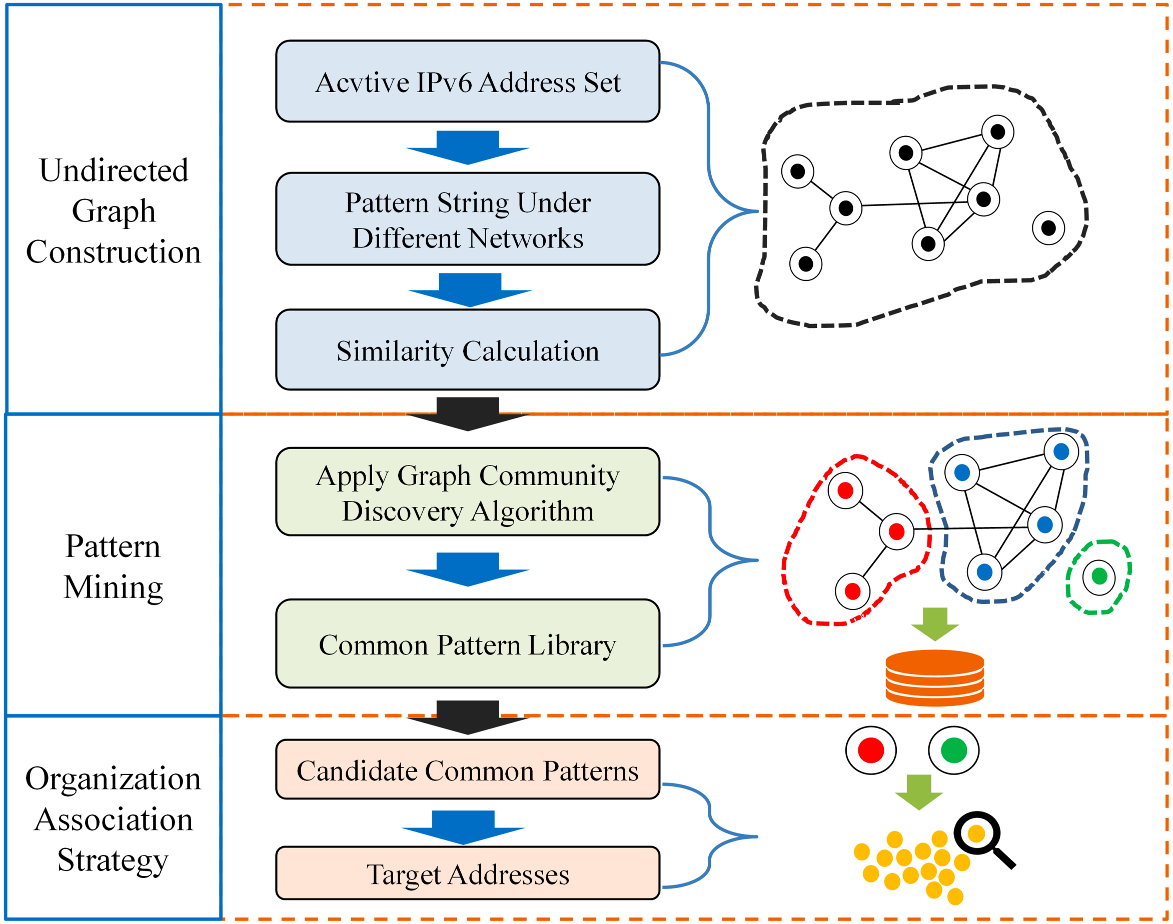
2001:dba8::8::1
2001:dba8::6::1
……
2003:3ef::1
2003:3ef::2

Commonality: more zeros in the high， and non-zero values in the low (Low bytes)

Core Ideas

Mine generic patterns and migrate to generate target addresses under any BGP prefix.

■ AddrMiner-N



Undirected Graph Construction

Acvtive IPv6 Address Set

Pattern String Under Different Networks

Similarity Calculation

Pattern Mining

Apply Graph Community Discovery Algorithm

Common Pattern Library

Organization Association Strategy

Candidate Common Patterns

Target Addresses

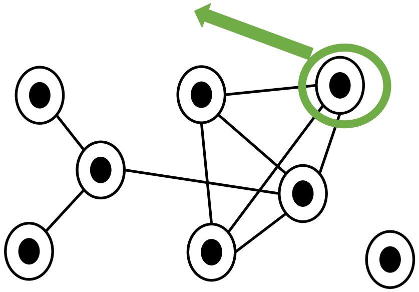
Workflow of AddrMiner-N

AddrMiner-N

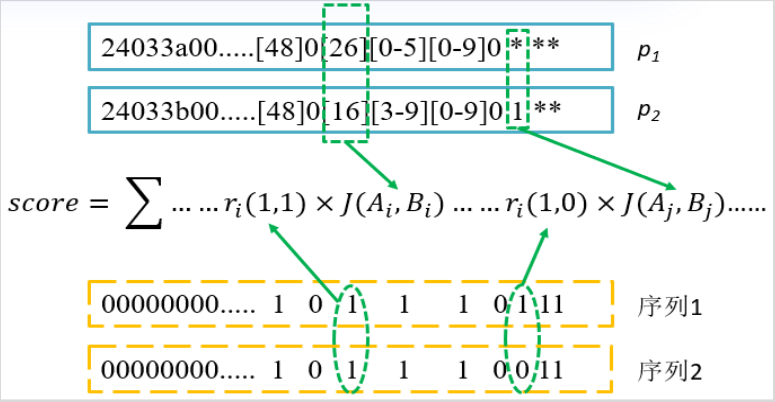
Undirected Graph Construction › **Pattern Mining** › **Address Detection**

1. The nodes of undirected graph represent the address patterns
2. The edges indicate the similarity of the different patterns
3. The weights represent the degree of similarity between different patterns

Address pattern. e.g. 2001:da8:*[0-8]:1



Undirected Graph Construction



24033a00.....[48]0[26][0-5][0-9]0 * ** p_1

24033b00.....[48]0[16][3-9][0-9]0 1 ** p_2

$$score = \sum \ldots\ldots r_i(1,1) \times J(A_i, B_i) \ldots\ldots r_i(1,0) \times J(A_j, B_j) \ldots\ldots$$

00000000..... 1 0 1 1 1 0 1 11 序列1

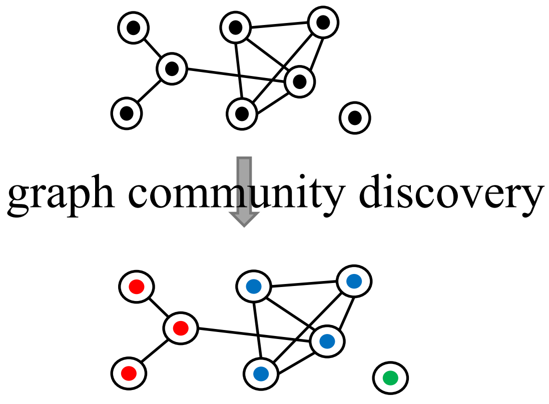
00000000..... 1 0 1 1 1 0 0 11 序列2

Calculation of the similarity of two patterns

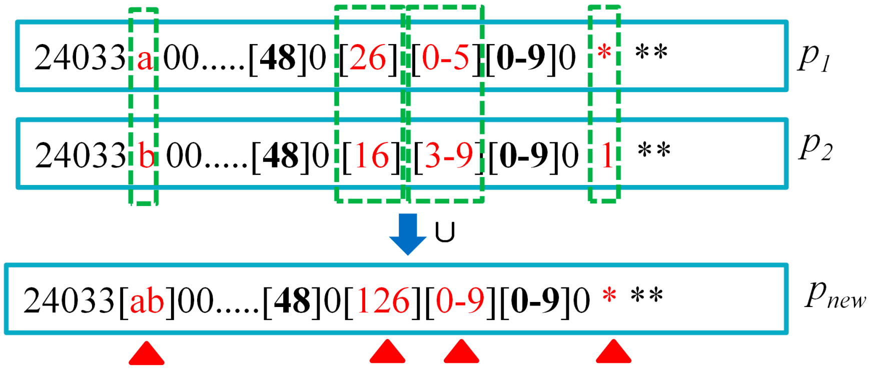
AddrMiner-N

Undirected Graph Construction › Pattern Mining › Address Detection

1. The graph community discovery algorithm will produce many communities
2. Merging pattern strings to build common pattern library



Pattern Mining



24033 a 00.....[**48**]0 [26] [0-5] [**0-9**]0 * ** p_1

24033 b 00.....[**48**]0 [16] [3-9] [**0-9**]0 1 ** p_2

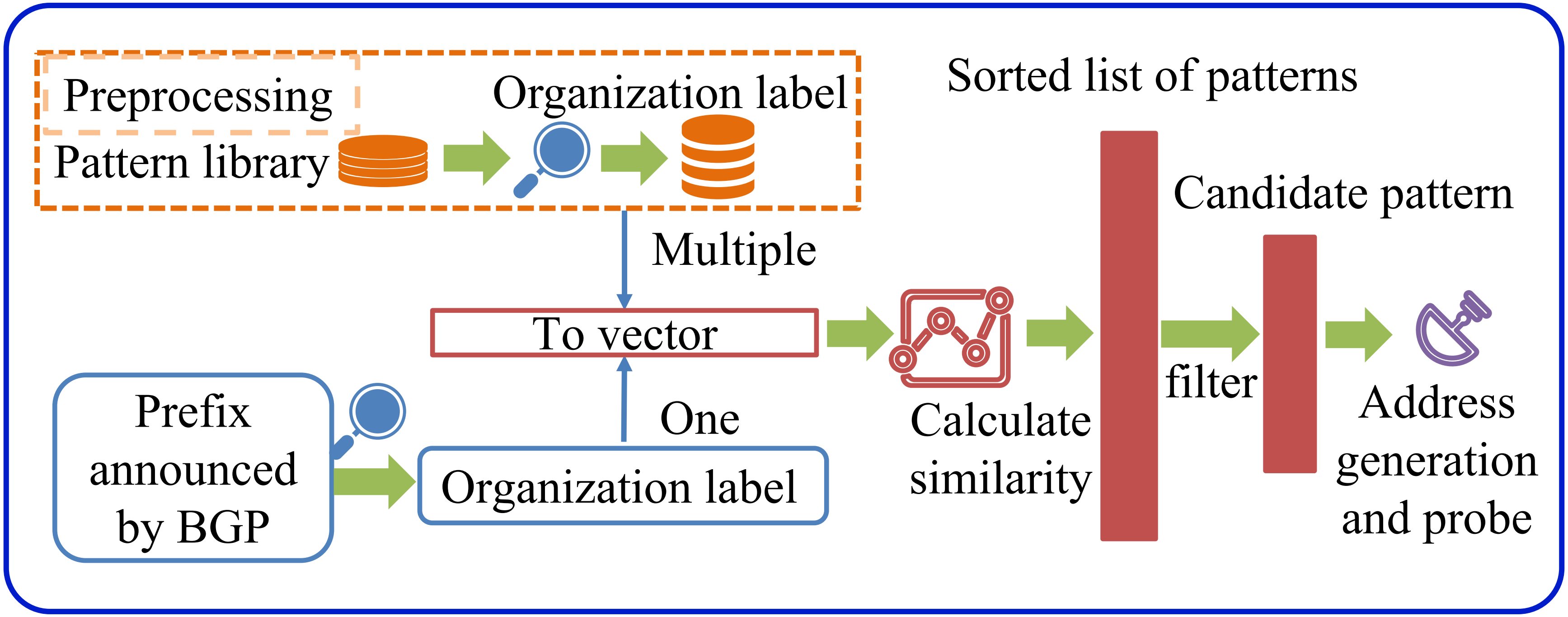
∪

24033[ab]00.....[**48**]0[126][0-9][**0-9**]0 * ** p_{new}

Merging process of different patterns

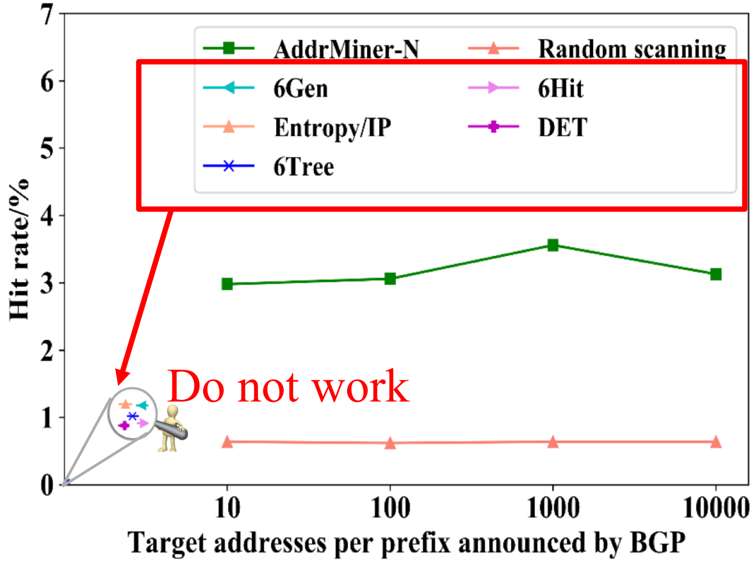
AddrMiner-N

Undirected Graph Construction › Pattern Mining › **Address Detection**



Organization association strategy

■ AddrMiner-N



Hit rate in the no seed scenario.

Table 2: Scenarios classification in the data set

Scenarios Classification	The number of BGP Prefixes
No seeds	56,730
Few seeds (≤ 10)	31,771
Sufficient seeds	17,472

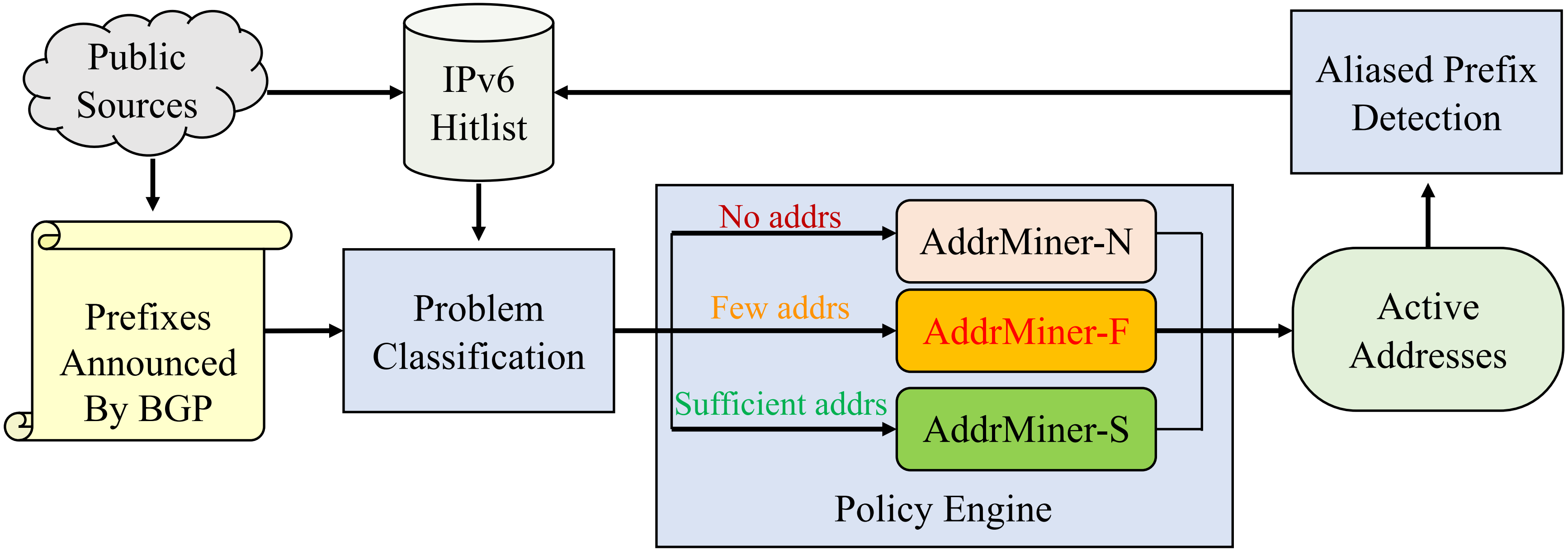
Table 3: The probing results of the two probing methods

Probing Method	#Active Addrs	#BPFXs	Coverage
AddrMiner-N	158,959,500	86,423	81.6%
Random Scanning	708,697	1,421	1.3%

BPFXs: BGP Prefixes.

Compared with existing solutions, AddrMiner-N is 60%-520% more efficient in detecting active IPv6 addresses, and the active addresses found cover more than 81% of BGP prefixes.

■ AddrMiner

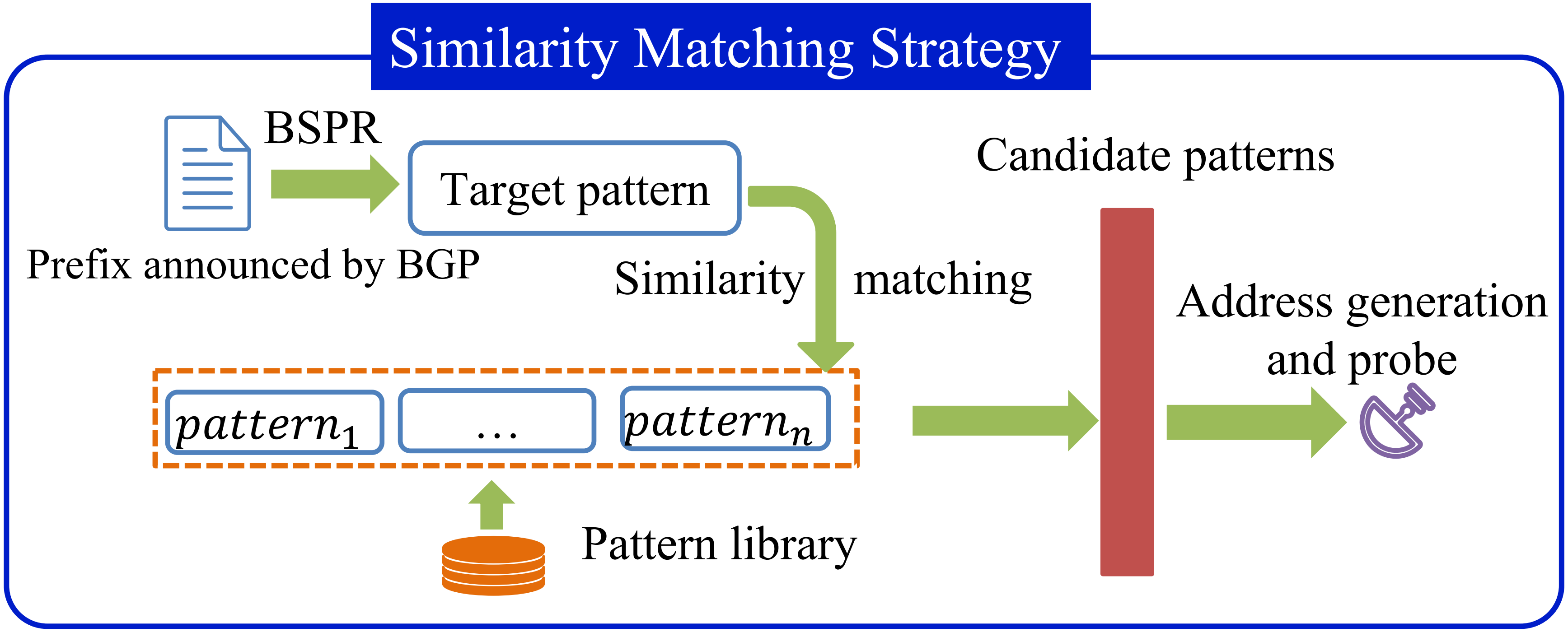


High-level overview of AddrMiner

AddrMiner implementation: https://github.com/AddrMiner/AddrMiner

■ AddrMiner-F

Undirected Graph Construction › Pattern Mining › **Address Detection**



Similarity matching strategy

■ AddrMiner-F

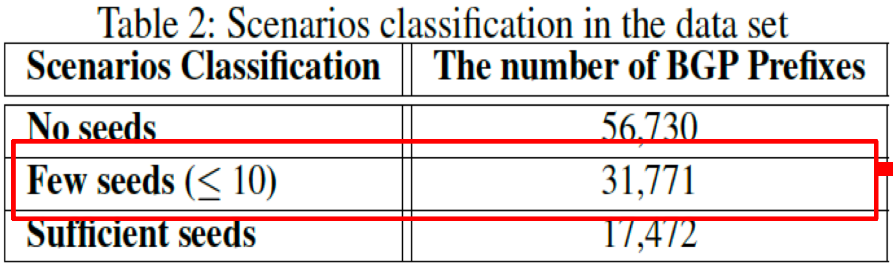
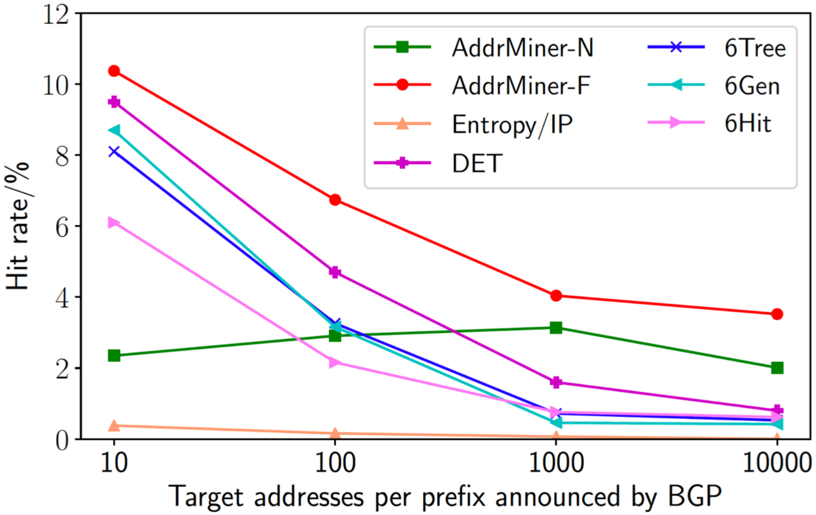


Table 2: Scenarios classification in the data set

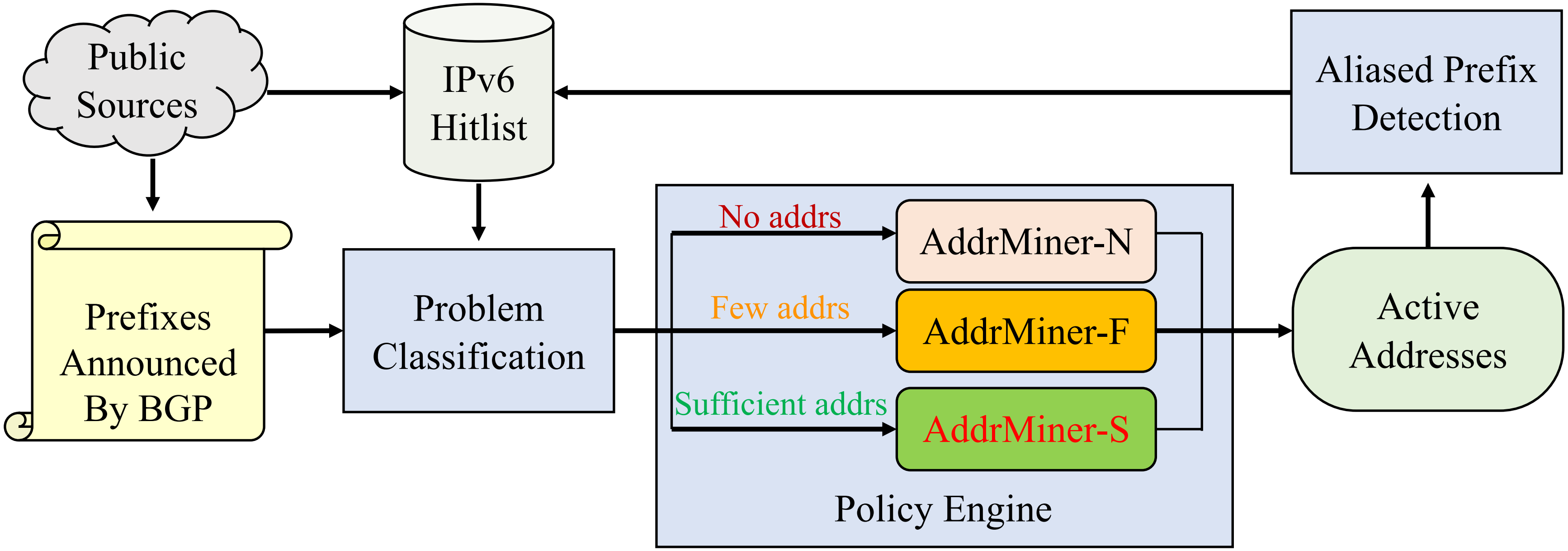
Scenarios Classification	The number of BGP Prefixes
No seeds	56,730
Few seeds (≤ 10)	31,771
Sufficient seeds	17,472



Hit rate in the few seed scenario.

Compared to existing solutions, AddrMiner-F is 70%-150% more efficient at detecting active IPv6 addresses.

AddrMiner



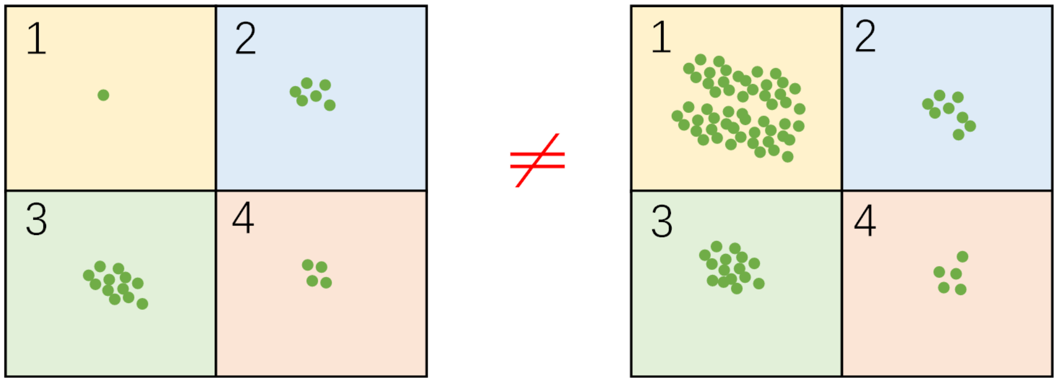
High-level overview of AddrMiner

AddrMiner implementation: https://github.com/AddrMiner/AddrMiner

■ AddrMiner-S

Assumption	The density of seeds is positively correlated with the density of real active IPv6 addresses

The sampling bias reduces the probing efficiency



Seed address density distribution

Active addresses density distribution in real network

● active addresses

Density distribution.

AddrMiner implementation: https://github.com/AddrMiner/AddrMiner

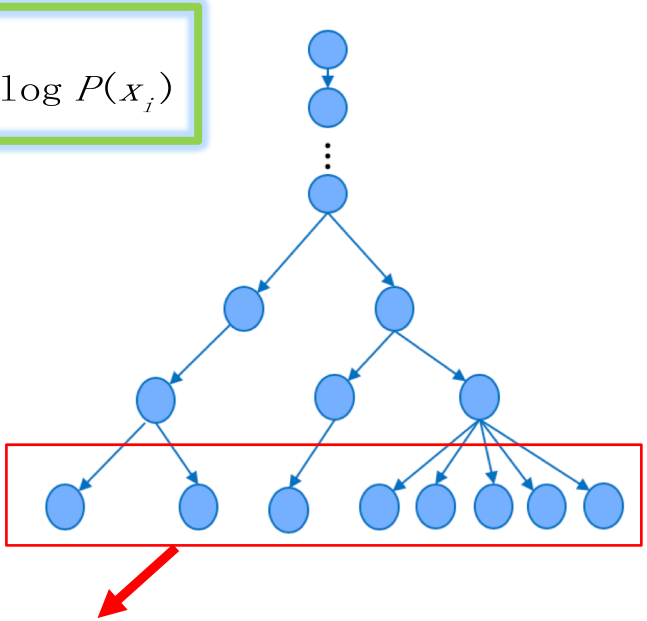
AddrMiner-S

- Discover high-density region of seed addresses

20010daf80000000000000010000000000000
20010daf80000000000000020000000000000
20010daf8100000000000003000000000f00
20010daf81000000000000080000000000000
20010daf81000000000000090000000000000
20010daf810000000000000a0000000000000
20010daf810000000000000b0000000000000
20010daf810000000000000c0000000000000

Entropy:

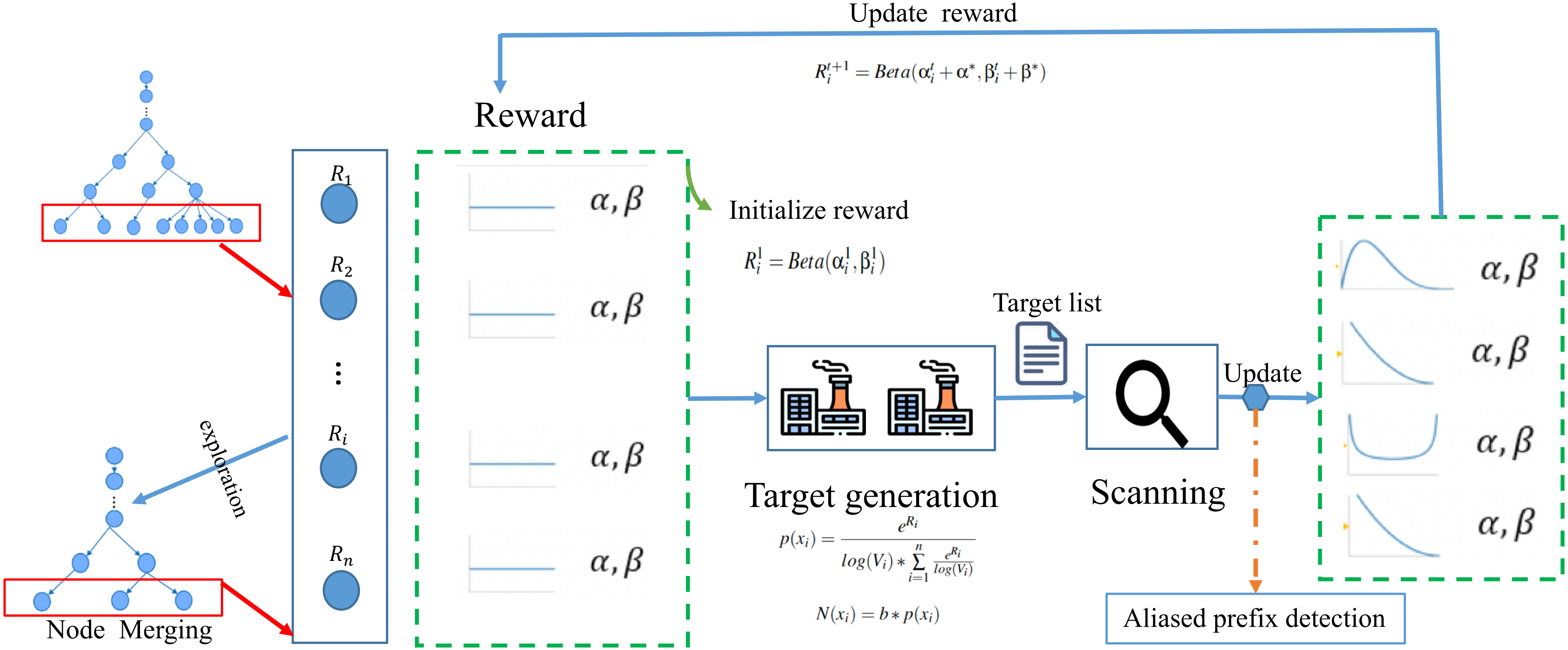
$$H(X) = -\sum_{i=1}^{k} P(x_i)\log P(x_i)$$



discover high-density regions of seed addresses

AddrMiner-S

- Target generation and update reward



$$R_i^{t+1} = Beta(\alpha_i^t + \alpha^*, \beta_i^t + \beta^*)$$

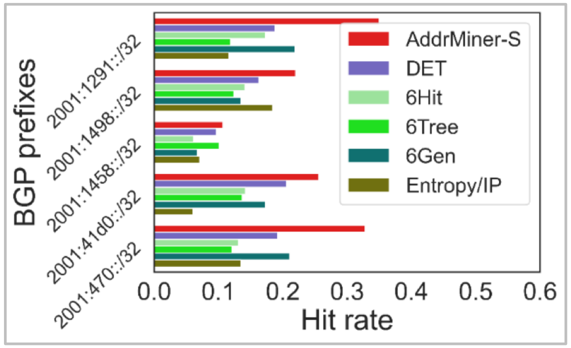
$$R_i^1 = Beta(\alpha_i^1, \beta_i^1)$$

$$p(x_i) = \frac{e^{R_i}}{log(V_i) * \sum_{i=1}^{n} \frac{e^{R_i}}{log(V_i)}}$$

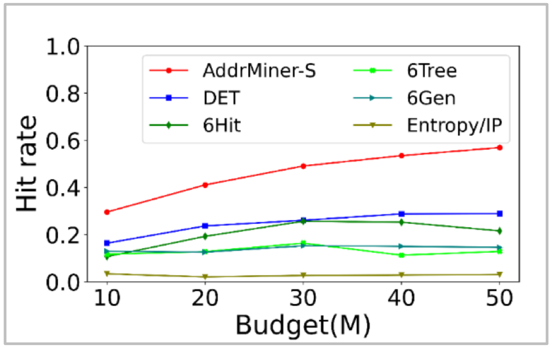
$$N(x_i) = b * p(x_i)$$

Workflow of AddrMiner-S

■ AddrMiner-S

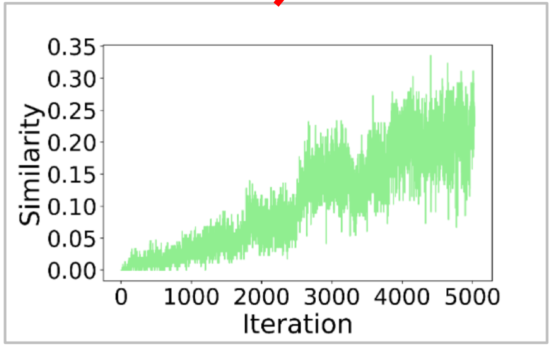


Hit rate in some prefixes



Hit rate in Gasser's hitlist

Eliminate sampling bias of seeds

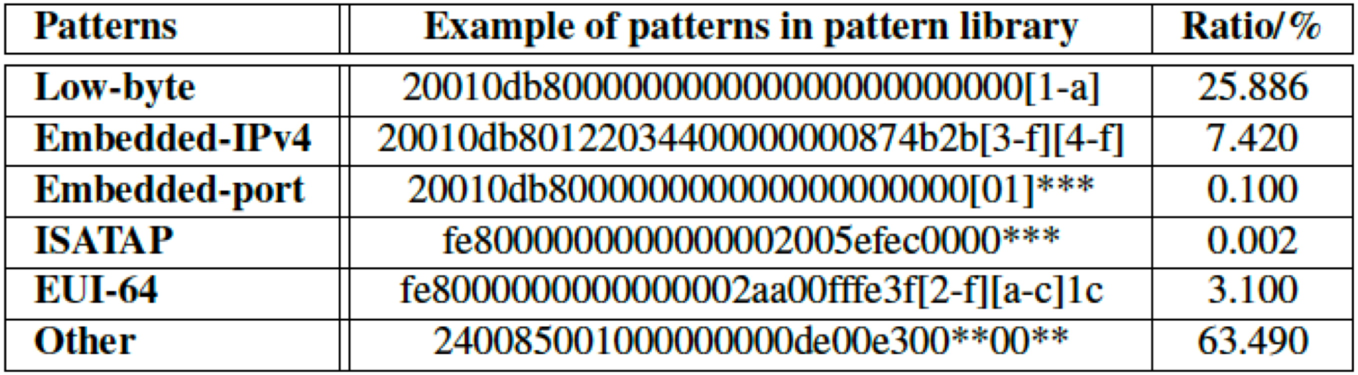


Consistency of density

Compared with existing solutions, AddrMiner-S has an active address hit rate of **56.3%** and a **94%-2000%** improvement when generating 50 million candidate addresses.

■ Pattern Library

Table 4: Ratio of common patterns in the pattern library



Patterns	Example of patterns in pattern library	Ratio/%
Low-byte	20010db800000000000000000000000[1-a]	25.886
Embedded-IPv4	20010db80122034400000000874b2b[3-f][4-f]	7.420
Embedded-port	20010db800000000000000000000[01]***	0.100
ISATAP	fe80000000000000002005efec0000***	0.002
EUI-64	fe80000000000000002aa00fffe3f[2-f][a-c]1c	3.100
Other	240085001000000000de00e300**00**	63.490

low-byte with a run of zeroes followed only by a low number; embedded-IPv4 inserting one IPv4 address embedded-port including the service port in the lowest-order byte of the IID; ISATAP IID with "0200:5EFE" flag and IPv4 address; EUI-64 IID with an embedded MAC address.

AddrMiner can dig out address patterns that not only contain the address patterns of RFC documents, but can also discover more valuable address patterns.

IPv6 Hitlist

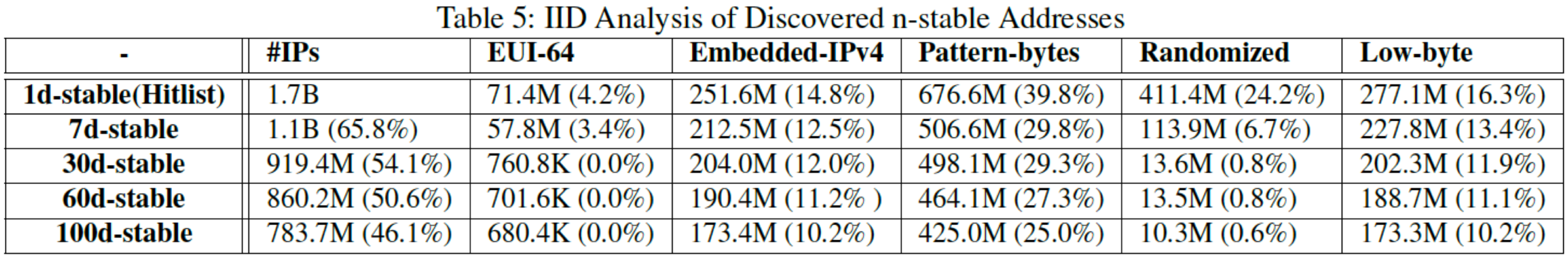


Table 5: IID Analysis of Discovered n-stable Addresses

-	#IPs	EUI-64	Embedded-IPv4	Pattern-bytes	Randomized	Low-byte
1d-stable(Hitlist)	1.7B	71.4M (4.2%)	251.6M (14.8%)	676.6M (39.8%)	411.4M (24.2%)	277.1M (16.3%)
7d-stable	1.1B (65.8%)	57.8M (3.4%)	212.5M (12.5%)	506.6M (29.8%)	113.9M (6.7%)	227.8M (13.4%)
30d-stable	919.4M (54.1%)	760.8K (0.0%)	204.0M (12.0%)	498.1M (29.3%)	13.6M (0.8%)	202.3M (11.9%)
60d-stable	860.2M (50.6%)	701.6K (0.0%)	190.4M (11.2%)	464.1M (27.3%)	13.5M (0.8%)	188.7M (11.1%)
100d-stable	783.7M (46.1%)	680.4K (0.0%)	173.4M (10.2%)	425.0M (25.0%)	10.3M (0.6%)	173.3M (10.2%)

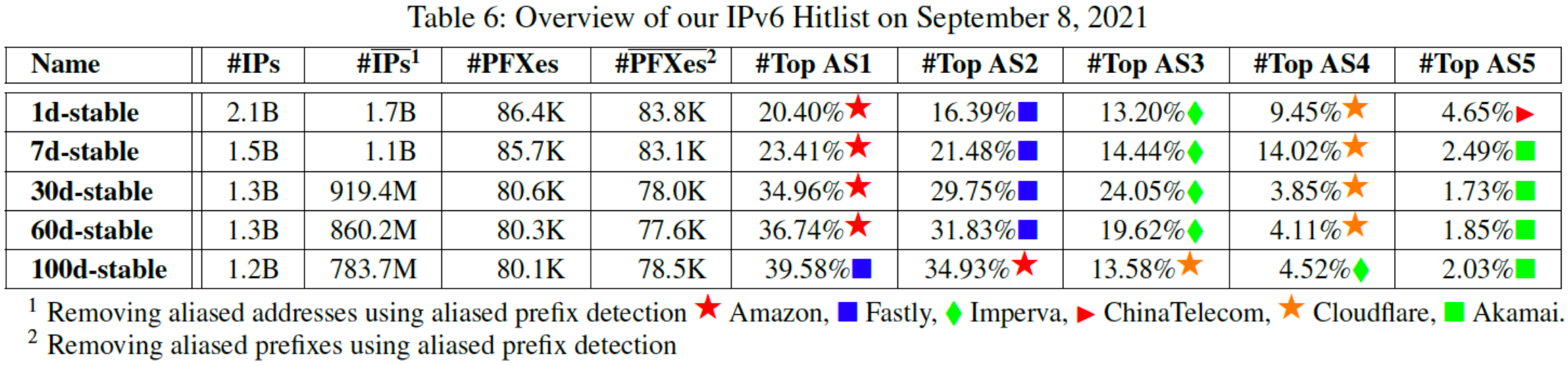


Table 6: Overview of our IPv6 Hitlist on September 8, 2021

Name	#IPs	#IPs[1]	#PFXes	#PFXes[2]	#Top AS1	#Top AS2	#Top AS3	#Top AS4	#Top AS5
1d-stable	2.1B	1.7B	86.4K	83.8K	20.40%★	16.39%🟦	13.20%◆	9.45%⭐	4.65%▶
7d-stable	1.5B	1.1B	85.7K	83.1K	23.41%★	21.48%🟦	14.44%◆	14.02%⭐	2.49%🟩
30d-stable	1.3B	919.4M	80.6K	78.0K	34.96%★	29.75%🟦	24.05%◆	3.85%⭐	1.73%🟩
60d-stable	1.3B	860.2M	80.3K	77.6K	36.74%★	31.83%🟦	19.62%◆	4.11%⭐	1.85%🟩
100d-stable	1.2B	783.7M	80.1K	78.5K	39.58%🟦	34.93%★	13.58%⭐	4.52%◆	2.03%🟩

[1] Removing aliased addresses using aliased prefix detection ★ Amazon, 🟦 Fastly, ◆ Imperva, ▶ ChinaTelecom, ⭐ Cloudflare, 🟩 Akamai.
[2] Removing aliased prefixes using aliased prefix detection

The IPv6 hitlist collected with greater quantity, higher quality, and wider distribution.

Will be publicly available: http://tsinghua-nmgroup-ipv6.cn/

■ Contributions

AddrMiner: A comprehensive global active IPv6 address probing system.

AddrMiner-N: filling the gap of address probing in the seedless address space regions

AddrMiner-F: More efficient active address detection algorithm in few seed regions

AddrMiner-S: More efficient active address detection algorithm in sufficient seed regions

IPv6 Hitlist: greater quantity, higher quality, and wider distribution

USENIX ATC '22

清華大學 Tsinghua University

Thanks for your attention!

Q&A

Email: sgl18@mails.Tsinghua.edu.cn