On Smartly Scanning of the Internet of Things

Jian Qu, Xiaobo Ma, *Member, IEEE*, Wenmao Liu, Hongqing Sang, Jianfeng Li, Lei Xue, Xiapu Luo, Zhenhua Li, *Senior Member, IEEE, ACM*, Li Feng, *Member, IEEE*, and Xiaohong Guan, *Life Fellow, IEEE*

***Abstract*— Cyber search engines, such as Shodan and Censys, have gained popularity due to their strong capability of indexing the Internet of Things (IoT). They actively scan and fingerprint IoT devices for unearthing IP-device mapping. Because of the large address space of the Internet and the mapping's mutative nature, efficiently tracking the evolution of IP-device mapping with a limited budget of scans is essential for building timely cyber search engines. An intuitive solution is to use reinforcement learning to schedule more scans to networks with high churn rates of IP-device mapping. However, such an intuitive solution has never been systematically studied. In this paper, we take the first step toward demystifying this problem based on our experiences in maintaining a global IoT scanning platform. Inspired by the measurement study of large-scale real-world IoT scan records, we land reinforcement learning onto a system capable of smartly scanning IoT devices in a principled way. We disclose key parameters affecting the effectiveness of different scanning strategies, and real-world experiments demonstrate that our system can scan up to around 40 times as many IP-device mapping mutations as random/sequential scanning.**

***Index Terms*— Internet of Things (IoT), adaptive algorithms, IP networks, cyberspace.**

I. Introduction

IN RECENT years, cyber search engines, such as Shodan [1], Censys [2], [3], and ZoomEye [4], have gained popularity among the security community due to their strong capability of indexing the Internet of Things (IoT) like webcams and routers. They actively scan IoT devices with fingerprints of various devices for unearthing IP-device mapping, offering publicly available search engine services. One can simply access these services using a browser and obtain IP-device mapping results by host names, IP addresses, certificates, or device-specific keywords. These search engines can also be used to effectively find IoT devices with certain (possible) vulnerabilities on the Internet [5], [6], [7]. In the short term, cyber search engines render attacks against IoT devices easier while offering a public channel for device owners (especially those with higher security requirements) to be informed of the exposure and protect the network from invaders accordingly. More importantly, in the long run, they would force IoT device manufacturers to make the best efforts to improve the security of their devices [8], [9].

We propose a method that can be used as a plug-in for existing cyber search engines to schedule scans to the large address space of the Internet smartly so that more IP-device mutations can be captured. Consequently, cyber search engines would become more timely. Because of the large address space of the Internet and the mutative nature of IP-device mappings, efficiently tracking the evolution of IP-device mapping with a limited budget of scans is essential for building timely cyber search engines. Specifically, our method learns the IP-device mutation intensity matrix through the scan record history. It assigns more scans to IP addresses that may have mutations in the next time slot to capture more mutations. Capturing more mutations is helpful for building a timely engine. High-rate scanning, in spite of the timeliness, usually induces excessive noises and thus may be blocked by firewalls. On the contrary, low-rate scanning is not noisy but not timely. As a result, the data obtained from cyber search engines would be generated a long time ago, rather than the up-to-date IoT devices on the Internet that are far more valuable.

Scanning IoT devices in a principled way to meet the timeliness requirement with a limited budget of scans depends on two major aspects. One is the resource investment, and the other is the scanning strategy. The former includes the number of servers, the servers' processing power, the bandwidth, etc. The latter concerns how to schedule scans encapsulating fingerprints of IoT devices across all IP addresses from the temporal perspective. Since the resource investment for cyber search engines is relatively stable, we only focus on the latter aspect. Existing cyber search engines, such as Censys, divide the protocol into different categories, and for each category, the scanning frequency is manually defined [3]. For example, Censys scans HTTP daily and SSH biweekly. Apparently, manually scheduling scans cannot maximize the timeliness of

Manuscript received 24 August 2022; revised 6 March 2023 and 10 July 2023; accepted 10 August 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. S. Sun. Date of publication 13 September 2023; date of current version 18 April 2024. This work was supported in part by the National Natural Science Foundation under Grant 61972313, Grant 62272381, Grant 62002306, Grant 62202405, and Grant T2341003; in part by the Natural Science Basic Research Program of Shaanxi Province under Grant 2023-JC-JQ-50; in part by the Fundamental Research Funds for the Central Universities, Postdoctoral Science Foundation, under Grant 2019M663725, Grant 2021T140543, and Grant 2023M732791; in part by the Hong Kong Research Grants Council (RGC) Project under Grant PolyU15223918; and in part by the CCF-NSFOCUS Kunpeng Research Fund of China. The work of Xiaobo Ma was supported by the Cyrus Tang Foundation. *(Corresponding author: Xiaobo Ma.)*

Jian Qu, Xiaobo Ma, Jianfeng Li, and Xiaohong Guan are with the MOE Key Laboratory for Intelligent Networks and Network Security and the Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: qj904154277@stu.xjtu.edu.cn; xma.cs@xjtu.edu.cn; jfli.xjtu@xjtu.edu.cn; xhguan@xjtu.edu.cn).

Wenmao Liu and Hongqing Sang are with NSFOCUS Inc., Beijing 100089, China (e-mail: liuwenmao@nsfocus.com; sanghongqing@nsfocus.com).

Lei Xue is with the School of Cyber Science and Technology, Sun Yat-sen University, Guangzhou 510275, China (e-mail: xuelei3@mail.sysu.edu.cn).

Xiapu Luo is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: csxluo@comp.polyu.edu.hk).

Zhenhua Li is with the School of Software and BNRist, Tsinghua University, Beijing 100084, China (e-mail: lizhenhua1983@gmail.com).

Li Feng is with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China (e-mail: fengli_xjtu@163.com).

Digital Object Identifier 10.1109/TNET.2023.3312162

cyber search engines because it does not consider IP-device mapping dynamics in different networks.

Despite its conceptual simplicity, how to design a system capable of smartly scheduling scans for IoT devices has never been systematically investigated. An intuitive solution is to use reinforcement learning to flexibly schedule more scans to networks with high churn rates of IP-device mapping, as can be learned from historical scanning records. However, there are two major challenges to applying reinforcement learning to designing the system. First, an immediate challenge is that designing such a system necessitates large-scale real-world IoT scanning records to have insights into IP-device mapping dynamics all over the Internet. However, no publicly available data is available for gaining insights and facilitating the design. Second, the IP-device mapping dynamics are driven by hidden factors that are hard to infer (e.g., which networks have the same IP assignment policy and can be characterized by similar mapping dynamics). This hurdle is further compounded by the Internet's tremendous and time-evolving nature, making it challenging to design the system.

To address these challenges, we carry out both measurement studies and system design for smartly scheduling scans for IoT devices. First, we perform large-scale measurements of IP-device mapping dynamics using our global IoT scanning platform. The measurement enables us to collect real-world IoT scanning records, quantify the IP-device mapping dynamics, and analyze factors affecting the dynamics. Second, inspired by the measurement study, we design a novel system that lands reinforcement learning onto guiding the smart scanning by exploiting the observation that the IP-device mapping dynamics in different networks may vary significantly.

Our system makes automatically learning IP-device mapping dynamics across different networks as a built-in feature for continuous scanning decision making, enabling the encouragement of scans to networks with high churn rates of IP-device mapping dynamic mapping and the discouragement of scans to those with low churn rates. To our best knowledge, we are the *first* to explore principled ways to scan IoT devices based on real-world measurement studies [10]. Our major contributions are summarized as follows.

- We perform measurements based on large-scale real-world IoT scanning records (consisting of 5,241,566 IP cameras) by scanning the entire IPv4 space for about 40 days, and quantify the IP-device mapping dynamics. The results reveal that both the IoT device types and IP address pools affect the dynamics.
- We land reinforcement learning onto a system capable of smartly scanning IoT devices. The system can encourage scans to networks with more dynamic IP-device mapping while impeding scans to those with less dynamic mapping. It consists of two novel strategies for scheduling scans based on online learning and batch learning. It could temporally schedule scans through continuous scanning decision making in consideration of historic IP-device mapping dynamics, and the hierarchically learned (spatial) IP address pools.
- Through extensive experiments, we demonstrate that our system could generally capture more IP-device mapping mutations than random and sequential scanning. We disclose the two key parameters affecting the effectiveness of different scanning strategies, i.e., the scan rate and the proportion of IoT devices to IP addresses. Real-world scanning experiments show that our system can scan up to around 40 times as many IP-device mapping mutations as random/sequential scanning.

Roadmap: Sec. II presents the literature. Sec. III performs measurement studies using real-world data. Sec. IV introduces system overview, Sec. V details system design, and Sec. VI reduces the computational complexity. Sec. VII conducts simulation-based evaluation and Sec.VIII performs real-world scanning experiments. Finally, we conclude in Sec. X.

II. Related Work

In the past few years, there has been a lot of research on Internet scanning. Salient scanning tools include Nmap, Zmap, Masscan, etc [3], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. Nmap is rich in functions, such as port scanning and device fingerprinting, but it is heavyweight and thus not suitable for large-scale scanning [18]. Zmap and Masscan use stateless scanning (e.g., no TCP three-way handshakes), achieving fast scanning [17].

Several studies focused on device identification [9], [18], [19], [22], [24], [25]. Nmap OS fingerprinting works by sending up to 16 specially designed packets to find the ambiguities in the standard protocol [18]. Miettinen et al. proposed a framework using software-defined networking for confining traffic of vulnerable devices [24]. Kohno et al. fingerprinted devices using clock skew in device hardware [19]. Feng et al. used automatically generated rules to inspect the application layer and identify devices [22]. Bezawada et al. used handcrafted features and Gradient boosting to classify device types [25]. Ma et al. leveraged machine learning methods to identify IoT devices by passively analyzing traffic [9]. Unlike these studies, our study focuses on the optimal scheduling of scans when actively probing devices for identification.

In the last decade, there have been cyber search engines using scanning technologies, such as port scanning, banner grabbing, and service fingerprinting [3], [15], [16]. Generally, these scanning technologies enable a cyber search engine to scan the IPv4 space periodically to collect information on connected devices, such as a device's hostname, IP address, open ports, and installed software. As the collected information grows, the cyber search engine can open a query interface on the Web to users using keywords like hostnames, IP addresses, and certificates. These engines help security practitioners discover, monitor, and analyze devices accessible from the Internet, thereby facilitating continuous monitoring of attack surfaces [2].

Published in 2009, Shodan provides information on about 500 million devices every month, including operating systems, hostnames, versions, and so forth [16]. Censys is another cyber search engine similar to Shodan [2]. It first uses ZMap to scan the entire IPv4 address space and then uses an application scanner called ZGrab to collect the handshake information of various protocols. Censys artificially defines protocol-dependent scanning frequencies and posts a timetable

of the scheduling strategy on its website [3], [17]. Some studies have indicated that Censys has a faster scanning and website update speed than Shodan [5], [26].

Cyber search engines can also help organizations identify and mitigate vulnerabilities in their systems. Durumeric revealed the presence of many vulnerable RSA and DSA keys due to the widely used insecure random number generators [13]. Genge et al. developed a vulnerability assessment function based on Shodan and disclosed 3,922 known vulnerabilities on 1,501 services. O'Hare et al. found 12,967 potential known vulnerabilities on 2,571 services [27]. Unlike existing studies, we focus on scanning scheduling strategies based on modeling IP-device mapping dynamics so to scan IoT devices in a principled way smartly. Our work complements existing studies and could be incorporated into a wide range of scanning tools.

III. Understanding IP-device Mapping Dynamics

A. Background

There are usually three ways to configure the IP address for a device, namely, Dynamic Host Configuration Protocol (DHCP), Point-to-Point Protocol (PPP), and static IP configuration [28], [29], [30]. Both DHCP and PPP will cause IP address changes frequently.

The DHCP server controls a pool of IP addresses. When connecting to the network, a client can be automatically assigned an IP address from the pool by the DHCP server. The client can keep the IP address within the lease duration (configured by the network manager). Upon the lease duration expiration, the client can send a message to the DHCP server to extend its lease for the same IP address [28]; if the client does nothing, the address will be revoked.

PPP can be encapsulated in data link layer protocols like PPP over Ethernet (PPPoE) and PPP over Asynchronous Transfer Mode (PPPoA). PPP first establishes a session between the client and the server. Then, the Internet Protocol Control Protocol (IPCP) is used to configure the client device's IP address. IPCP does not have a lease duration. The IP address is released when the PPP session ends [29], [30], [31].

In both DHCP and PPP, if the session of the device continues, the IP address will not change. IP address mutations can be triggered by both the client and the server. On the client side, once the device re-establishes the PPP session, a new IP address will be assigned to the device; if a DHCP client is offline for a period of time that exceeds the lease duration and then reconnects the DHCP server, its IP address will change. On the server side, Padmanabhan et al. found that some ISPs may limit the session duration (typically a multiple of 24 hours), and the IP address will change periodically [32].

B. Measuring IP-Device Mapping Dynamics

To understand IP-device mapping dynamics, one needs to perform large-scale scans encapsulating fingerprints of IoT devices globally and collect detailed scanning records. Despite the prevalence of IoT fingerprinting techniques [18], [19], [22], no public scanning record is available to facilitate the study.

To this end, we take the first step to measure real-world IP-device mapping dynamics by performing large-scale scanning campaigns using our globally deployed (commercial) IoT scanning platform. Specifically, we scanned the entire IPv4 space to identify IP cameras, the most popular type of IoT device on the Internet. The scanning campaign was repeated four times, starting on June 5, June 15, June 26, and July 6, 2021, respectively. The scanning campaign lasted about 10 days at each time, resulting in 2,896,824, 3,089,436, 3,093,510, and 3,076,343 successful scanning records, respectively.

Note that the traffic fingerprints of these IP cameras are extracted from their banner text that commonly describes device types explicitly. We have been manually maintaining and labeling a database of IoT fingerprints with the aid of machine learning. For more details, the reader can refer to [33].

The four scanning campaigns allow us to measure IP-device mapping mutations by comparing scanning records. Suppose we perform a scanning campaign at time t_1 and the result is S_{t_1}, the set of (successfully scanned) mapping between IP addresses and device types. For example, we assume that $S_{t_1} = \{(\alpha_1, \text{“D-Link Camera”}), (\alpha_2, \text{“TVT Camera”})\}$, where α_1 and α_2 are two IP address. S_{t_2} is the result of the scanning campaign at time t_2, and $S_{t_2} = \{(\alpha_1, \text{“D-Link Camera”})\}$. We employ Jaccard similarity to measure the mapping mutations between t_1 and t_2:

$$J(t_1, t_2) = \frac{|S_{t_1} \cap S_{t_2}|}{|S_{t_1} \cup S_{t_2}|}. \tag{1}$$

In this example, $|S_{t_1} \cap S_{t_2}|$ equals 1, $|S_{t_1} \cup S_{t_2}|$ equals 2, and the Jaccard similarity equals 0.5. If there are no IP-device mapping mutations between two scans, $J(t_1, t_2)$ will equal 1. As the mutations become significant, $J(t_1, t_2)$ will approach 0. Note that, when $J(t_1, t_2)$ equals 1, we cannot be 100% sure that there are no mapping mutations, due to the possibility of an IP address being re-assigned to a device of the same type as the device that initially owns the IP address. However, the probability of mapping mutations would be extremely small and ignorable in such a case.

1) Device Types: In our scanning campaigns, we successfully find a cumulative number of 5,241,566 IP cameras. Fig. 1 shows the average number of devices of the top 10 popular IP camera types across all campaigns. For a specific device type, we calculate the mapping mutations between every two successive campaigns using (1). Consequently, three values of Jaccard similarity measuring mapping mutations are derived, and we plot the average value in Fig. 1. It can be seen that the Jaccard similarity between two successive scanning campaigns differs significantly across IP camera types. For example, the Jaccard similarity of the Samsung camera is about 0.83, but that of the Hipcam camera is less than 0.45. This indicates that the IP-device mapping dynamics are device-type-specific.

2) IP Pools: We consider an IP address pool as a set of IP addresses (possibly) under the same IP management policy. Different IP address pools may have different IP-device mapping mutation intensities because each has its configuration, capacity, and occupancy. Padmanabhan et al. found that the

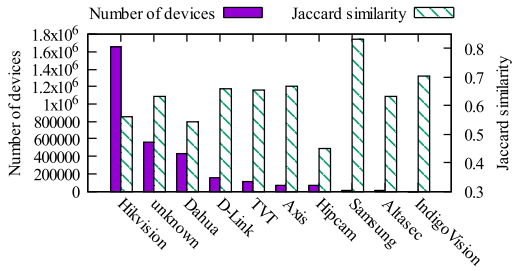


Fig. 1. The number of cameras from different manufacturers and Jaccard similarity between the two scans.

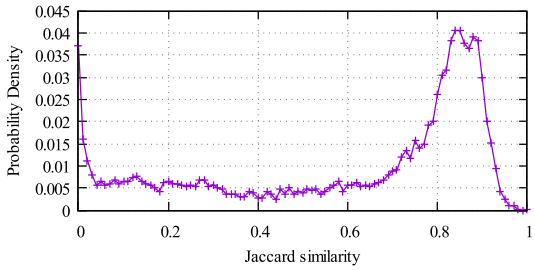


Fig. 2. The probability distribution of the Jaccard similarity between two scans across all IP address pools.

frequency of IP address changes was related to geographic location [32]. In other words, the IP-device mapping mutations are related to the IP address pools.

To gain (coarse-grained) insights into IP-device mapping mutations of large-sized IP pools, we define an IP address pool as a class B IP space, and the IPv4 space is divided into 65,536 IP address pools. For each pool, we derive (1) between every two adjacent scanning campaigns and calculate the average Jaccard similarity. Our calculation neglects the IP address pools with less than 50 identified IP cameras for statistical validity. Fig. 2 presents the probability distribution of the average Jaccard similarity across all IP address pools. Although there are two peaks when the Jaccard similarity approaches 0.85 or slightly exceeds 0, the overall distribution is dispersed. This implies that the IP-device mapping dynamics depend on IP address pools.

IV. System Overview

Since the IP-device mapping dynamics are related to device types and IP address pools, we exploit this observation to design a system capable of smartly scheduling scans for IoT devices. Fig. 3 shows the architecture of our proposed system.

First, a *priority matrix* describing the IP-device scanning tasks is fed into the system. Each matrix element is a priority value (i.e., 1, 2, 3, ...) specifying the order to execute the corresponding scanning task defined by (device type, IP address). Initially, the priority matrix is randomly or sequentially defined. As the scanning proceeds, we will collect more historical records consisting of (IP, device, last scan time). Then, we derive a *probability matrix* quantifying the probability of IP-device mutation. Each element of this matrix denotes the probability that the corresponding IP-device mapping mutations in the following scheduled scan. We keep updating the probability matrix as new scanning records arrive, while simultaneously returning probability ranking as feedback to refresh the priority matrix. Finally, the tasks with

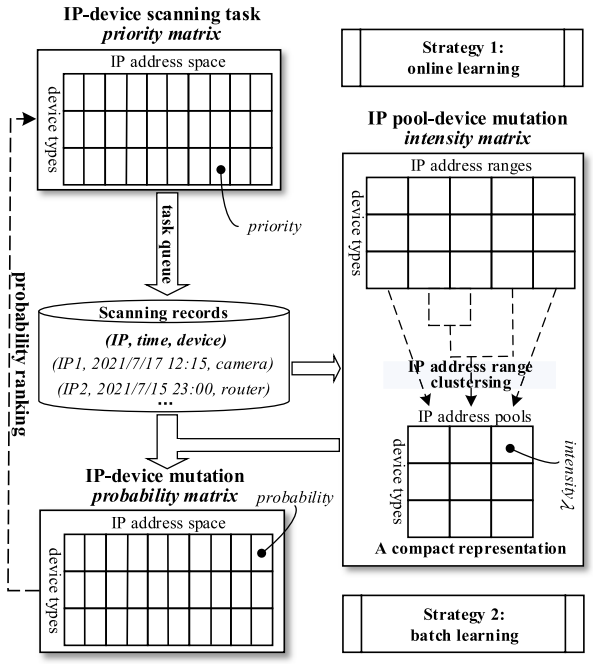


Fig. 3. The architecture of the proposed IoT scanning system.

larger values of mutation probability would be assigned higher priorities in the task queue.

When we derive the probability matrix, not only do we use scanning records but also an *intensity matrix*. The rationale is that the IP-device mutation probability in the following scheduled scan depends on both temporal and spatial factors. The temporal factor is the time interval between the current time and the last scan time (in scanning records). As the time interval increases, the mutation probability grows because of occurrences of events such as device replacement, and IP reconfiguration. The spatial factor is the intensity matrix that characterizes the overall likelihood of IP-device mutation in individual IP address pools. Each pool is expected to be under the management of the same IP assignment policy, and hence devices in that range are statistically coherent in terms of IP-device mutation. IP-device mappings belonging to IP address pools with stronger mutation intensity tend to have higher mutation probability.

The intensity matrix is estimated using scanning records. Initially, each IP address pool is defined as the IP address range of a small network, say a class C network, to ensure its high probability under the management of the same IP assignment policy. As scanning records accumulate, our system will hierarchically cluster small IP address ranges into large IP address pools, resulting in a compact representation of the intensity matrix. The compact representation leads to a more computationally effective estimation of the intensity matrix. More importantly, as the size of an IP address pool increases, the estimation accuracy for the intensity matrix increases due to the growing number of scanning records in that pool.

The above architecture of our system constitutes a continuous scanning decision-making process, which is a natural fit for reinforcement learning algorithms. Model-free methods and model-based methods are two types of reinforcement learning. Compared with model-based methods, model-free

TABLE I
SUMMARY OF MAJOR NOTATIONS

Notation	Definition
d	a device type
r	an IP address range
A	the set of scanning tasks
$\lhd(t_1,d_1),(t_2,d_2)\rhd$	a 2-gram scanning record
$R(d)$	all 2-gram scanning records with $d_1 = d$
S_t	the set of the latest scanning records at time t
$\pi(S_t)$	the next scheduled scanning tasks given S_t
λ	the IP pool-device mapping mutation intensity

methods like Deep Q-Network [34] are not sample efficient, resulting in a slow convergence rate. Consequently, they are not suitable for the scanning environment where slow convergence fails to capture the fast mutations. Therefore, we propose two model-based reinforcement learning-based scanning strategies under the designed architecture, namely, online learning and batch learning, according to the way to build the intensity matrix. The online learning strategy incrementally updates the intensity matrix while performing scanning and directly enters the continuous scanning decision making process. The batch learning strategy, however, proactively scans abundant information to build the intensity matrix before entering the continuous scanning decision making process.

V. SYSTEM DESIGN

Following the proposed architecture, we detail our system design. First, we present the design goal and system model. Then, we design online learning and batch learning scanning strategies, and the IP address pool estimation technique. Table I summarizes major notations in our system.

A. Designing Goal

We model the IoT scanning process as a continuous decision-making process. For a certain type of IoT device, the basic task of the decision-making is to select one IP address, scan it, and get the result. That is, the task can be regarded as a cycle of selection and scan. The cycle will be repeated continuously, hence constituting the entire scanning process.

From the perspective of IoT devices, our goal is to optimize the scanning strategy to capture as many IP address changes as possible. Such a goal is equivalent to capturing as many IoT device changes as possible from the perspective of IP addresses. To sum up, we aim to capture as many IP-device mapping mutations as possible. Note that we treat an IP address without hosting any device as a special IP-device mapping. During the scanning process, we score the reward as one upon capturing a mutation. Given a fixed number of scans and a period of time, we aim to maximize the total reward.

B. System Model

1) Scanning Process Modeling: The scanning can be modeled as a process of continuously making decisions on the priorities of IP-device scanning tasks. All the scanning tasks are organized in the priority matrix. At a high level, the scanning process keeps refreshing the priority matrix regularly after executing a bunch of scanning tasks. The refreshing is actually to fine-tune the priority values based on the set of the latest scanning records of all IP addresses.

Let $S_t(t = t_0, t_1, t_2, \ldots)$ be the set of the latest scanning records of all IP addresses at time t, and A denotes the set of scanning tasks. Then, the problem becomes fine-tuning the priority value for each scanning task in A based on S_t. Intuitively, scanning tasks with larger IP-device mutation probabilities (during the following scheduled scan) should be assigned higher priorities. More precisely, given S_t, top-k scanning tasks in terms of mutation probability ranking, denoted by $\pi(S_t)$, join the queue of the following scheduled scan. $\pi(S_t)$ is formally expressed as

$$\pi(S_t) = \{a|\text{if } P(a|S_t) \in \text{top-k}(P(a|S_t)), a \in A\}, \quad (2)$$

where $P(a|S_t)$ is the IP-device mapping mutation probability when we choose a to scan at state S_t. Apparently, modeling IP-device mapping mutation and estimate the value of $P(a|S_t)$ is crucial in the scanning process.

We would like to point out that, in real-world scanning, scanning tasks associated with one IP address but multiple device types could be performed simultaneously to improve the scanning efficiency, especially when the device types share the same port. For example, many types of commercial IP cameras open TCP port 554 by default. In this case, a scan using a single TCP connection destined to TCP port 554 of the target IP address will identify the camera type.

2) IP-Device Mapping Mutation Modeling: Consider an event as one IP-device mapping mutation. Naturally, for a certain device, the event arrivals can be modeled as a non-homogeneous Poisson process [35]. The reasons are twofold. First, the events of a device's IP address changes are generally independent of each other. For example, in the DHCP configuration mentioned in Sec. III-A, the IP address changes are independent since they are attributed to many random factors, such as device online-offline dynamics and human intervention; in the static IP configuration, the two typical events, namely, initial IP assignment and final IP release, are also independent. Second, in different periods of time, the rate of IP address changes is likely to be different. In other words, the rate of IP address changes is time-dependent. Formally, the event arrivals have the following properties.

$$\begin{cases} P[N(t+\Delta t) - N(t) = 1] = \lambda(t)\Delta t + o(\Delta t), \\ P[N(t+\Delta t) - N(t) \geq 2] = o(\Delta t), \end{cases} \quad (3)$$

where $\lambda(t)$, a non-negative function of time t and device type d, denotes the mapping mutation intensity (i.e., the rate of IP address changes at t), and $N(t)$ represents the number of IP address changes during $(t, t+\Delta t]$, of a certain device.

Suppose t_1 is the last time to scan the IP address a, and the current time is t_2. At t_1, we find that a hosts a device d, which is recorded in the state s. Assume that the mapping mutation intensity of device d is $\lambda(t)$. Then, the probability of capturing the IP-device mapping mutation at t_2 is approximated as

$$P(a|S_t) = 1 - e^{-\int_{t_1}^{t_2} \lambda(t)dt}. \quad (4)$$

The reason why it is approximate rather than strictly equal is as follows. After the device's IP address changes, other devices of the same type may be assigned the IP address again with a very low (almost ignorable) probability.

Recall that both the IoT device types and IP address pools affect the IP-device mapping dynamics, as is revealed in Sec. III. We maintain and estimate $\lambda(t)$ separately for each pair of device type and IP address pools in the intensity matrix demonstrated in Fig. 3.

3) Mapping Mutation Intensity Estimation: If mapping mutation intensity, i.e., $\lambda(t)$, can be estimated, the mutation probability $P(a|S_t)$ would be calculated. We can estimate $\lambda(t)$ using historic scanning records. Maximum likelihood estimation and least-squares methods can be used for parameter estimation. However, they are challenging to solve.

Let us take maximum likelihood estimation as an example. In the simplest case, $\lambda(t)$ is a constant, which means λ is independent of time and only related to the device type. We define a *2-gram scanning record* as a (successive) subsequence of the scanning record sequence of an IP address. A 2-gram scanning record can be expressed as $<(t_1,d_1),(t_2,d_2)>$, where t_1 is the scanning time, t_2 is the scanning time following t_1, d_1 is the device type at t_1, and d_2 is the device type at t_2. The probability that a 2-gram scanning record occurs is

$$P(<(t_1,d_1),(t_2,d_2)>|\lambda)=\begin{cases}e^{-\lambda(t_2-t_1)} & \text{if } d_1{=}d_2,\\ 1{-}e^{-\lambda(t_2-t_1)} & \text{if } d_1{\neq}d_2.\end{cases} \quad (5)$$

λ is the IP-device mapping mutation intensity of device type d_1. Let $R(d_1)$ denote the set of every 2-gram scanning record with the first device type equal to d_1 across all IP addresses. The likelihood function of d_1 changing to other device types across all IP addresses is

$$L(\lambda|R(d_1)) = \prod_{r\in R(d_1)} P(r|\lambda). \quad (6)$$

Then, we maximize the likelihood function to estimate λ. However, the maximal value of this function is difficult to solve by calculating the zero point of the derivative, even if the log-likelihood function is used. To make things exacerbated, λ may not be a constant, and parameter estimation would be challenging. We need a solution to estimate λ.

Padmanabhan et al. revealed that the address changing intensity depends on the time during a 24-hour day [32]. Therefore, we model $\lambda(t)$ as a periodic function with a period of 24 hours for each device type. To estimate this continuous function, we can discretize it into small intervals. Specifically, we divide $\lambda(t)$ into 24 intervals, and the problem becomes estimating 24 discrete variables. We use $T(\lambda(t))$ to represent any one of the intervals, i.e., $[0,1)$, $[1,2)$,..., and $[23,24)$. We notice that the probability $\mathrm{P}(d_1 \neq d_2|T(\lambda) \subseteq (t_1,t_2])$ grows as $\lambda(t)$ increases. We use this probability to approximate $\lambda(t)$.

Next, we design two strategies for estimating $\lambda(t)$: the online learning strategy and the batch learning strategy.

C. Online Learning Strategy

The online learning strategy incrementally updates the estimation of $\lambda(t)$ as the scanning records arrive. The strategy is detailed in Algorithm 1.

In Algorithm 1, $\lambda(d,r,t)$ is represented by a three-dimensional array. The first dimension is the device type, the second dimension means the IP address range, and the third dimension refers to the 24 discrete time intervals. r_a represents the IP address range index of address a. $\lambda(d,r,t)$ equals $(y(d,r,t)+1)/(y(d,r,t)+n(d,r,t)+1)$ in the algorithm, where $y(d,r,t)$ and $n(d,r,t)$ represent the number of times the scanning records change and does not change within the time period of $T(\lambda(d,r,t))$, respectively.

Algorithm 1 Scanning Using Online Learning Strategy

Input: scanning task set
$A = \{\text{IP addresses}\} \times \{\text{device types}\}$
Output: scanning records
initialization:
$\lambda(d,r,t) = y(d,r,t) = n(d,r,t) = 0$;
$S_t(a) = (t_0, d_0)$ for each $a \in A$;
while *True* **do**
  $\lambda(d,r,t) = \frac{y(d,r,t)+1}{y(d,r,t)+n(d,r,t)+1}$;
  scan $\pi(S_t)$ and get a set of 2-gram scanning records E;
  for *each* $[a, <(t_1,d_1),(t_2,d_2)>]$ *in* $[\pi(S_t), E]$ **do**
    if $d_1 \neq d_2$ **then**
      $y(d_1,r_a,t)$ += $\frac{|T(\lambda(d,r_a,t))\cap(t_1,t_2]|}{t_2-t_1}$;
    else
      $n(d_1,r_a,t)$ += $\frac{|T(\lambda(d,r_a,t))\cap(t_1,t_2]|}{|T(\lambda(d,r_a,t))|}$;
    end
    update $S_t(a) = (t_2, d_2)$;
  end
end

Specifically, for a record $<(t_1,d_1),(t_2,d_2)>$, if $d_1 \neq d_2$, $|T(\lambda(d,r_a,t)) \cap [t_1,t_2)|/(t_2-t_1)$ is added to $y(d,r_a,t)$, where $|T|$ refers to the time duration of T. This means that we split the contribution of one scanning record into multiple time intervals, since the mutation could happen at any time interval between t_1 and t_2. Similarly, $|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|/|T(\lambda(d,r_a,t))|$ is added to $n(d,r_a,t)$ if $d_1 = d_2$, because no mutation between t_1 and t_2 in the scanning record indicates no mutation in all time intervals between t_1 and t_2.

The reason why we make $\lambda(d,r,t)$ equal to $(y(d,r,t)+1)/(y(d,r,t)+n(d,r,t)+1)$ instead of $y(d,r,t)/(y(d,r,t)+n(d,r,t))$ is to balance exploration and exploitation in online reinforcement learning [36]. Our solution is a combination of *upper confidence bound* and *Laplace smoothing*. In our solution, all the values of $\lambda(d,r,t)$ are equal to one at the beginning of the scan, thereby allowing us to explore different IP addresses at the initial period randomly. When the time tends to infinity, $\lambda(d,r,t)$ gradually stabilizes.

D. Batch Learning Strategy

In the online learning strategy, when the average scanning interval of each IP address exceeds 24 hours, the value of $\lambda(d,r,t)$ will be very similar, making the segmentation of time meaningless. For example, suppose we have a 2-gram record $<(t_1,d_1),(t_2,d_2)>$, Algorithm 1 works well when t_2-t_1 is small. However, if t_2-t_1 surpasses 24 hours, the contribution of this 2-gram record to estimating $\lambda(t)$ will be limited.

To address the limitation of the online learning strategy, we propose the batch learning strategy in Algorithm 2. The

Algorithm 2 Scanning Using Batch Learning Strategy

Input: scanning task set
$A = \{\text{IP addresses}\} \times \{\text{device types}\}$
Output: scanning records
initialization:
$\lambda(d, r, t) = y(d, r, t) = n(d, r, t) = 0$;
$S_t(a) = (t_0, d_0)$ for each $a \in A$;
—Stage 1: batch learning
while *True* **do**
  perform sequential scanning and get $< (t_1, d_1), (t_2, d_2) >$;
  if $d_1 \neq d_2$ **then**
    $y(d_1, r_a, t)$ += $\frac{|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|}{t_2 - t_1}$;
  else
    $n(d_1, r_a, t)$ += $\frac{|T(\lambda(d,r_a,t)) \cap (t_1,t_2]|}{|T(\lambda(d,r_a,t))|}$;
  end
  update $S_t(a) = (t_2, d_2)$;
end
$\lambda(d, r, t) = \frac{y(d,r,t)+1}{y(d,r,t)+n(d,r,t)+1}$;
—Stage 2: delayed scanning
while *True* **do**
  calculate $P(a|s)$ for each address a using (4);
  scan address a' that maximize $P(a'|s)$;
end

basic idea is to conduct a special scan to collect information (i.e., batch learning), and then use the collected information to perform scanning (i.e., delayed scanning). We efficiently and proactively collect needed information before entering a continuous decision-making scanning process.

In Algorithm 2, the entire scanning process is divided into two stages. Stage 1 performs batching learning. Specifically, we set a fixed interval for sequential scans to estimate $\lambda(t)$ efficiently. The estimation algorithm is the same as that in Algorithm 1, except that we calculate $\lambda(t)$ after Stage 1 (rather than at the time of every scan). Stage 2 conducts delayed scanning. We calculate $P(a|s)$ for each IP address and scan the IP address with the highest value of $P(a|s)$. Note that $\lambda(t)$ does not change any longer once Stage 2 starts.

The advantage of batch learning scanning is the capability of accurately estimating $\lambda(t)$. Therefore, the final scanning performance could be improved, yet at the cost of extra scanning investment at the batching learning stage.

E. IP Pool Estimation Based on IP Address Range Clustering

After several rounds of scanning, the IP address range clustering algorithm can be used to estimate the IP address pools, resulting in a compact representation of the intensity matrix. During the scanning process, we gradually obtain statistics about IP-device mapping dynamics. Besides helping us perform scans more efficiently, such statistics allow us to estimate IP address pools. If some addresses have the same characteristics, they are likely to belong to the same IP address pool. The key to this problem is to find some features so that the features of the IP address ranges in the same pool are as similar as possible, and the features of the IP address ranges in different IP address pools differ from each other.

An effective feature can be device distribution. Given a certain IP address pool, although the IP addresses of different devices may change internally, the population distribution of different types of devices will keep stable. Moreover, for different IP address pools, the device distribution will be different. Therefore, the device distribution can be exploited to identify IP address pools. Since it is easy to know which IP addresses are statically configured by keeping track of the evolution of IP-device mapping, we filter out all static IP addresses while estimating IP address pools.

To exploit device distribution for estimating IP address pools, we employ a clustering-based method. Specifically, we use 256 IP addresses as the smallest unit of IP address ranges to perform clustering since many network administrators also consider the IP address range comprising 256 addresses as the smallest unit. For each block of IP addresses, we calculate the population distribution of different types of devices using the scanning records and cluster different IP address ranges based on the distances of the population distribution. We use hierarchical clustering to perform the clustering. The reason is that hierarchical clustering does not require pre-defining the number of clusters.

VI. Optimizing System Scalability

In Sec. V, we have designed online and batch learning strategies for the smart scanning system. However, directly deploying these two strategies makes the system both compute and storage intensive, thereby making it difficult to scale up in large-scale scanning of a number of IoTs. The major reason is that these strategies rely on regularly calculating the probabilities of mutations for each task (to select the tasks of top priority) in large-scale scanning (e.g., the entire IPv4 space).

A. Bucket-Based Online Learning Strategy

To make our system scale up, we reduce the computation and storage overhead by designing a bucket-based online learning strategy. As Fig. 4 demonstrates, the strategy divides the rates of performing scanning tasks into discrete levels, and for each scan rate level, a bucket that contains a queue of tasks is maintained. A queue of tasks in a bucket is executed at the corresponding (constant) scan rate level. Initially, all the tasks are in the same bucket with the same (initial) scan rate level. Upon the execution of any task, if an IP-device mapping mutation occurs, the task will be probabilistically moved into a higher bucket with a larger scan rate level, and if no mutation occurs, the task will be probabilistically moved into a lower bucket with a smaller scan rate level.

A group of tasks of top priority would be selected from all the buckets under the constraints of maximum task execution cache, the number of tasks in each bucket, and the scan rate levels. Let us consider three buckets with scan rate levels of $R_1 = 1$, $R_2 = 2$, and $R_3 = 3$, respectively. Suppose the numbers of tasks in the three buckets are all equal to 100, and the maximum task execution cache can only process 10 tasks.

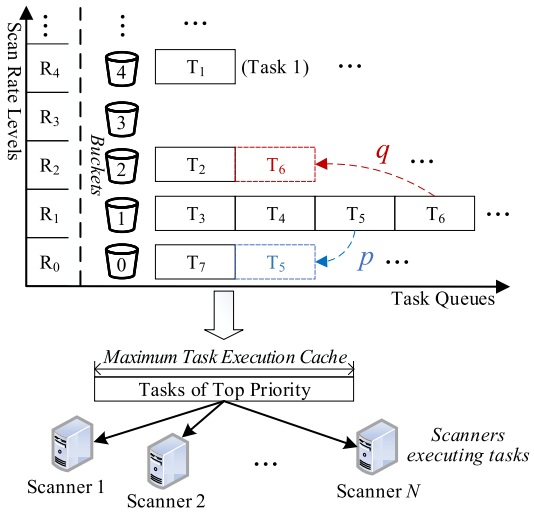


Fig. 4. A bucket-based online learning strategy for scalable scanning.

In this case, the tasks in the three buckets are selected in a ratio of 1:2:3 (i.e., $1\times100 : 2\times100 : 3\times100$) to add up to 10 tasks. Suppose the numbers of tasks in the three buckets are 300, 150, and 100, respectively. Then, the tasks in the three buckets are selected in a ratio of 1:1:1 (i.e., $1\times300 : 2\times150 : 3\times100$). It can be proved that, if the scan rate level of each task is accurately estimated, the selected tasks are also the ones with the largest IP-device mutation probabilities.

The bucket-based online learning strategy allows one to only calculate the probabilities to update (increase or decrease) the scan rate levels of those just executed tasks, instead of regularly calculating the probabilities of IP-device mutations for all the tasks (e.g., all IPv4 addresses) and selecting the top-ranked tasks in terms of mutation probabilities. Therefore, the computational and storage overhead is drastically reduced.

B. Probabilistic Scan Rate Update

Probabilistic scan rate update aims at dynamically adjusting the scan rate levels of the just executed task based on the scanning results. The basic idea is to move the task to a bucket with a higher scan rate level once an IP-device mapping mutation occurs when executing the task; conversely, if there is no mutation, the task would be moved to the bucket with a lower scan rate level. For example, in Fig. 4, the task denoted by T_6 is moved to R_{102} with a *rising probability* q conditioned on an IP-device mutation, and the task denoted by T_5 is moved to R_{100} with a *falling probability* p if no mutation occurs.

Recall that all the tasks are initially in the same bucket with the same scan rate level. Because of the inborn differences of IP-device dynamics across tasks, the hitting probabilities (i.e., the probabilities of finding an IP-device mutation) of executing different tasks depend on specific tasks and the scan rate level. For a particular task, say T, we denote its hitting probability at the scan rate level R_i by $r_T(R_i)$.

Once T falls into the tasks of top priority (i.e., those with top-ranked hitting probabilities) and gets executed, the probability of moving T to the higher scan rate level equals $r_T(R_i)q$, and moving T to the lower scan rate level happens with a probability of $(1-r_T(R_i))p$. If $r_T(R_i)q > (1-r_T(R_i))p$, T moves upward, the scan rate R_i increases to R_{i+1}, and accordingly $r_T(R_i)$ decreases to $r_T(R_{i+1})$; and if $r_T(R_i)q < (1-r_T(R_i))p$, T moves downward, the scan rate R_i decreases to R_{i-1}, and accordingly $r_T(R_i)$ increases to $r_T(R_{i-1})$. Therefore, we can adjust the scan rate levels of different tasks by carefully setting the values of q and p.

Finally, $r_T(R_i)q$ and $(1-r_T(R_i))p$ become equal and T stabilizes at a certain scan rate level. Suppose that task T is expected to stabilize at the optimal scan rate level R_*. In this case, we have

$$r_T(R_*)q = (1-r_T(R_*))p. \tag{7}$$

In addition, since we select the tasks with top-ranked hitting probabilities to execute, the value differences of $r_T(R_*)$ across tasks lead to the moving upward/downward of the executed tasks. Therefore, the optimal scan rate level R_* should make the values of $r_T(R_*)$ (i.e., the hitting probabilities) *equal* across all the tasks so that our strategy could converge.

Let us denote the above equalized hitting probability across all the tasks by $\bar{r}$, which is the average hitting probability of all the tasks.

Substituting $r_T(R_*)$ in (7) by $\bar{r}$, we have

$$p = \frac{\bar{r}}{(1-\bar{r})}q. \tag{8}$$

This equation can be used to determine the values of p and q.

Algorithm 3 summarizes our strategy. It accepts a set of tasks and a rising probability q as the input. During the initialization stage, it establishes 256 buckets and puts all the tasks in the middle bucket. During the scan cycle, we execute a batch of tasks and get a set of scanning records E using the method mentioned in Sec. VI-A. Then, we update the average hitting probability $\bar{r}$ with the scanning records E, and update the scan rate of each executed task.

Using the probabilistic scan rate update algorithm, one can adjust the scan rate following the real mutation intensity. For example, if scanning an IP pool is suddenly prohibited by a firewall, the mutation intensity of the tasks in this pool will become 0. The scanner will gradually reduce the scan rate of the tasks in this pool until the removal of firewall blocking.

In the NAT case, multiple devices may share the same IP address. If the device uses port mapping, the scanner can still discover it. Otherwise, the scanner may not receive any response. Our method is also applicable when multiple devices share the same IP address using different ports. Specifically, in the real-world scan (Section VI-C), one task can be recognized as a tuple of an IP address and the target port, such as $< IP : 100.100.100.100, port : 21 >$. For the same IP address, different ports will not interfere with each other.

C. Tunable System Overhead

The bucket-based online learning strategy allows one to tune the two parameters below to control the system overhead within affordable resources.

1) The Number of Buckets: Fewer buckets mean fewer scan rate levels and less computation and storage overhead. To make the limited number of scan rate levels cover a large range of scan rates, we use the exponential growth method

Algorithm 3 Probabilistic Scan Rate Update

Input: scanning task set $A = \{a_0, a_1, a_2, \ldots, a_n\}$ and tasks,
rising probability q
Output: scanning records
Initialization:
$R_{0-255} = \emptyset$, $R_{127} = A$;
$S_t(a) = (t_0, d_0)$ for each a;
Scan Cycle:
while *True* **do**
  perform scanning and get a set of 2-gram scanning records E;
  update $\bar{r}$;
  for *each* $< (t_1, d_1), (t_2, d_2) >$ *in* E **do**
    if $d_1 \neq d_2$ **then**
      calculate p using (8);
      move to a lower bucket with probability p;
    else
      move to a higher bucket with probability q;
    end
    update $S_t(a) = (t_2, d_2)$;
  end
end

to generate the scan rate levels. Specifically, for any bucket R_i, the scan rate of R_{i+1} is equal to $a \times R_i$, where a is a constant greater than 1 ($a = 1.05$ in our experiments). With this method, only hundreds of buckets are needed.

2) Partition Granularity of the Task Set: A partition of the task set consists of many task subsets. Each task subset normally consists of scanning targets relating to each other. Each task subset could be considered a basic task that constitutes the minimum scheduling unit. Such a basic task may be a specific one, like scanning one IP address for a particular device, or a broad one, like scanning a range of IP addresses for many devices. Broader task partition granularity reduces scheduling complexity due to the decreased total number of tasks.

Setting partition granularity is task-specific. Below we introduce three typical granularities to partition a task set A into task subsets.

3) IP-Port Granularity: This is the finest granularity. One task subset is recognized as a tuple of an IP address and the target port, such as $< IP : 100.100.100.100, port : 21 >$, wherein this subset would be assigned a personalized scan rate level.

4) IPs-Port Granularity: A range of IP addresses and a certain port is recognized as a task subset. In the experiments, we normally consider a class C IP space and a port as a task subset, e.g., $< IPs : 100.100.100.0 - 255, port : 21 >$.

5) IPs-Port-Device Granularity: On the basis of the IPs-port granularity, the device type can be added to differentiate the scan rate levels of different devices in the same network identified by the IPs-port. As such, an example of a task subset is $< IPs : 100.100.100.0 - 255, port : 21, device : a\ hash\ value\ denoting\ Amazon\ Echo >$. In real-world scanning, the device type may be agnostic. However, in the context of figuring out scanning policies, we do not need to identify the type of each device accurately, but only to distinguish between different devices. For example, we can extract fields from the scan results like banners, and then use the hash function to distinguish between different devices.

D. Scheduling Complexity Analysis

The scheduling complexity of the bucket-based online learning strategy is attributed to two processes, namely, scan rate update and selection of tasks of top priority.

In the scan rate update process, given a set of 2-gram scanning records, for each record, one needs to find the task relating to the record and update the task state to determine whether an IP-device mutation occurs. Since it only needs $O(1)$ time complexity to find the task relating to the record, The overall time complexity of this process is $O(n)$ (n is the number of records). This process also requires saving the previous scanning records for state comparison. To this end, a hash table can be used. Suppose the average hitting rate is $r (0 < r < 1)$, and the size of the task space is n, and the total space complexity is $O(rn)$.

The task selection process selects top-n-ranked tasks in terms of their priorities. To figure out the top-n-ranked tasks, the strategy traverses all of the buckets, hence taking a time complexity of $O(l)$, where l represents the number of buckets. The output time cost is proportional to the number of tasks, and the time complexity is $O(n)$. Therefore, the total time complexity in the task selection process is $O(n+l)$. The space required by this process lies in preserving the data structure in Fig. 4. The data structure contains l buckets and n tasks. If we use a linked-list structure to store the tasks, the total space complexity is $O(n + l)$.

VII. Simulation-Based Evaluation

We first simulate the scenario where many devices change their IP addresses in large-scale networks. The simulation enables us to exactly set various parameters and discover potential factors influencing the performance. It also allows us to use random seeds to ensure that different scanning strategies work in precisely the same simulation environment for a fair comparison.

A. Experiment Settings

The simulation environment is built based on two components. One is the simulation of individual devices. The other is the simulation of large-scale autonomous networks. In the simulation of individual devices, we make the IP-device mapping mutations as a nonhomogeneous Poisson process and $\lambda(t)$ is a periodic function with a period of 24 hours for each device. For each device, we assume that the IP-device mapping mutations occur instantaneously, and the target IP address that a device transfers to is randomly selected in the remaining IP addresses of the IP address pool where the device resides.

Note that an autonomous network may have multiple IP address pools, and each pool has its own configuration. For simplicity, we assume that all devices' IP addresses are dynamically assigned. This assumption does not hinder the practical

TABLE II
DEFAULT PARAMETER SETTINGS

Parameters Name	Value
Total number of IP addresses	8,192 (32 class C networks)
Total number of IP pools	10
Scan rate	0.5 IP addresses per second
The proportion of devices to addresses	0.8
Number of device types	20
Number of scanning rounds	100
λ in Scenario A	$1/\lambda \sim$ U(0h,24h)
Address change time in Scenario B	$t \sim$ U(0h,24h)

use of our strategy, as it is easy to know which IP addresses are statically configured. To be more realistic, for each IP address pool, the IP address assignment is set continuously, and the number of IP addresses is a multiple of 256.

We set the number of devices and the device distribution differently across IP address pools. We randomly generate the number of each type of device for each IP address pool, and randomly assign an initial IP address for each device. Under these settings, we mainly test two typical scenarios below:

Scenario A: All the IP-device mapping mutations are subject to the *homogeneous* Poisson process. For each combination of device type and IP address range, we assign a random λ.

Scenario B: All the IP-device mapping mutations follow a *non-homogeneous* Poisson process. Each type of device changes its address at a certain time of each day.

Scenario A can be considered the worst-case scenario for our scanning strategies due to the maximized randomness of IP-device mapping dynamics (hence little information to exploit), while scenario B is a more general and best-case scenario. Accordingly, the experiments under these two scenarios can represent the proposed strategies' lower and upper-bound performance, respectively.

Table II details our parameter settings, including the total number of IP addresses, the total number of IP pools, the scan rate, etc. We define a round of scan as scanning all addresses once, and the scanning will end after a certain number of rounds. Note that both λ and t in Table II depend on device types and IP address ranges.

B. Scanning Performance Using Different Strategies

To evaluate the performance of our online learning scanning and batch learning scanning, we compare them with naive strategies, including random scanning and sequential scanning. In addition, we use a "God's view scanning" strategy, which can know the specific time when a device changes its IP address in Scenario B. Therefore, the God's view strategy can be used as the upper bound of all the scanning strategies.

To eliminate the influence of the absolute number of IP-device mapping mutations and make a fair comparison, we define the relative performance (RP) score to measure the performance of each strategy. The RP score is defined as

$$\text{RP score} = \frac{N_i(t)}{N_r(t)}, \tag{9}$$

where $N_i(t)$ and $N_r(t)$ represent the number of IP-device mapping mutations at time t using scanning strategies i and "random scanning", respectively. Particularly, when i is the random scanning, RP score (relative performance of a strategy over random scanning) is always equal to 1.

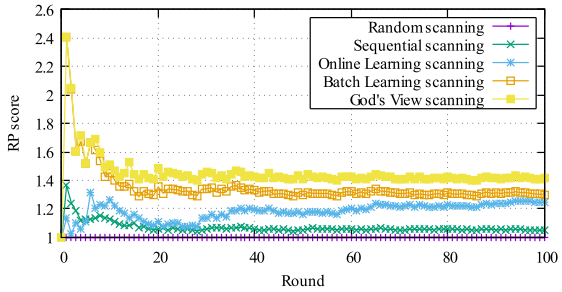


Fig. 5. A scanning example using different strategies in Scenario B. The default parameters in Table II are used.

Fig. 5 shows an experiment using different strategies in Scenario B. In this experiment, for a fair comparison, we set the number of IP-device mapping mutations of each device to be the same across different strategies. The x-axis represents the round number of scanning, and one round of scanning means that the strategy finishes scanning all IP addresses. The y-axis represents the RP score of each strategy.

In Fig. 5, we see that the curve is not stable at the beginning, especially within the first ten rounds, because of the significant variance caused by insufficient samples. As the round number increases, the curve becomes more stable. At the 100th round when the scan ends, the RP scores of the strategies "God's view scanning", "batch learning scanning", "online learning scanning", "sequential scanning" and "random scanning" are 1.42, 1.30, 1.24, 1.05, and 1.00, respectively.

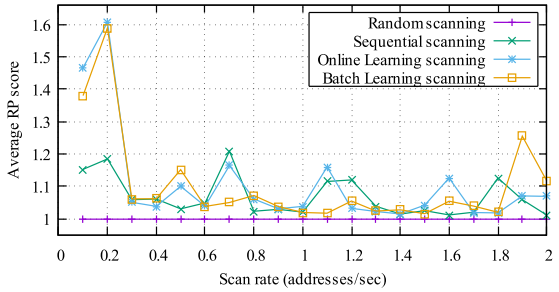
Among all scanning strategies other than the God's view, the batch learning scanning performs the best. However, such performance comes at the cost of conducting 100 rounds of scanning to collect information (i.e, the batch learning stage) before entering the continuous decision-making-based scanning. The RP score of the online learning scanning increases as the scanning proceeds. In summary, the experiment shows that our strategies significantly outperform random scanning and sequential scanning under the parameters in Table II.

C. Performance Sensitivity

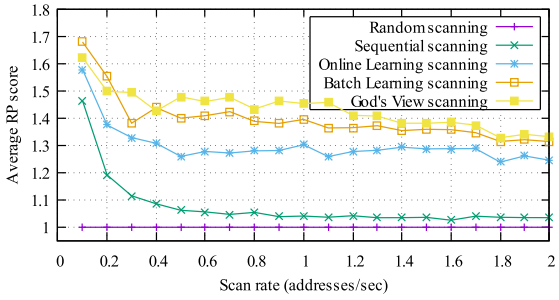
Previous experiments are conducted using the parameter settings in Table II. To gain insight into the performance under different parameter settings, we examine the impact of different parameter settings on scanning strategies.

By varying a certain parameter and keeping the remaining parameters constant, we find that there are mainly three parameters that significantly impact the RP score of different strategies, namely, the scan rate, the proportion of devices to IP addresses, and the proportion of specially-configured IP addresses. To understand the influence of these parameters on the scanning performance, we conduct experiments, and the results are shown in Fig. 6, Fig. 7, and Fig. 8.

Consider that, even if each experiment is conducted with the same parameters, the varying λ of different device types may result in fluctuations in the experiment results. Therefore, we use the average results of multiple experiments. Each point in Fig. 6, Fig. 7 and Fig. 8 represents the average RP score across 100 experiments with the same parameters.

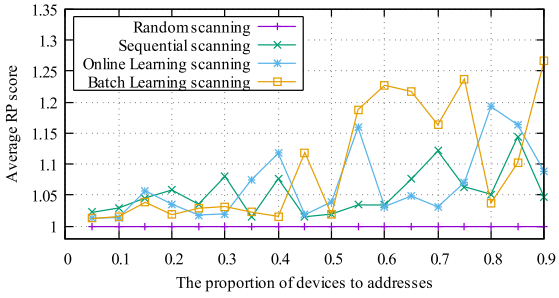


(a) Scenario A

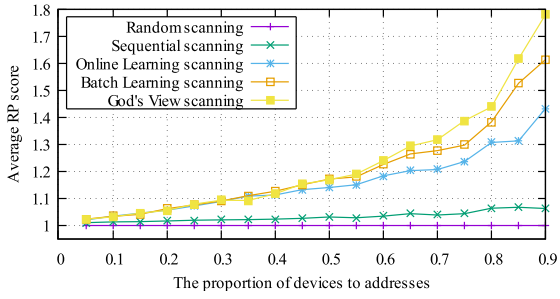


(b) Scenario B

Fig. 6. Average RP score under different scan rates. Each data point is the average of 100 experiments under the same parameters.



(a) Scenario A

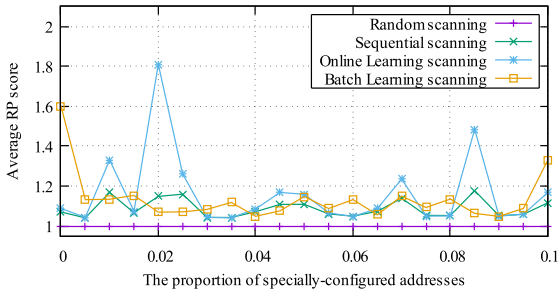


(b) Scenario B

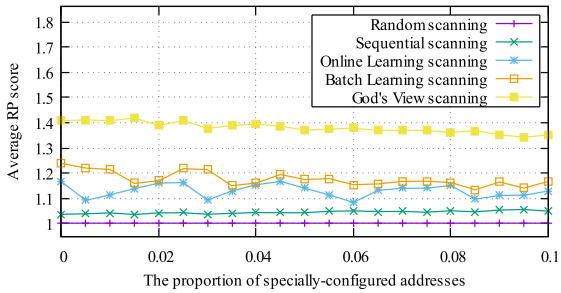
Fig. 7. Average RP score over varying proportions of devices to addresses. Each data point is the average of 100 experiments under the same parameters.

In each experiment, the scan is conducted for 100 rounds. In Scenario A, we compare four scanning strategies and five strategies in Scenario B. Note that the God's view strategy only works in Scenario B.

1) Scan Rate: Fig. 6(a) and Fig. 6(b) represent the performance of scanning strategies with different scan rates in Scenario A and Scenario B. We find that, as the scan rate increases, the average RP score gradually decreases. This indicates that as the scan rate tends to be positive infinity, all IP-device mapping mutations will be captured by any strategy.



(a) Scenario A



(b) Scenario B

Fig. 8. Average RP score over varying proportions of specially-configured addresses. Each data point is the average of 100 experiments under the same parameters.

The temporal fluctuation of the average RP score in Scenario B is smaller than that in Scenario A. Meanwhile, the average RP score improvement that our proposed scanning strategies make over random scanning in Scenario B is higher than that in Scenario A. We believe the reason is that the IP-device mapping mutations in Scenario A are more uncertain than that in Scenario B, thereby restricting the capability of our strategies. Particularly, when the scan rate is small, the performance improvement over sequential scanning is not significant. However, when the scan rate grows large, the performance improvement becomes significant.

2) The Proportion of Devices to IP Addresses: Fig. 7(a) and Fig. 7(b) depict the performance using different proportions of devices to IP addresses in Scenario A and Scenario B. In these two figures, the temporal fluctuation of the average RP score in Scenario A is still larger than that in Scenario B.

As the proportion of devices to IP addresses increases, the absolute number of IP address mutations captured by all strategies grows larger, but the growth rates of different strategies differ from each other. Specifically, the growth rates of our strategies are larger than those of random and sequential scanning. This implies that, as the number of IoT devices on the Internet grows, our strategies would become far more advantageous than random and sequential scanning.

3) The Proportion of Specially-Configured IP Addresses: IP address relocation dynamics, attributed to IP address relocation policies, can impact the performance of scanning strategies. Normally, a pool of neighboring IP addresses (e.g., those belonging to a class C network) follows the same IP address relocation policy since they tend to be in the same administrative domain. However, some IP addresses in a pool of neighboring IP addresses may not follow the same IP address relocation policy, and we term

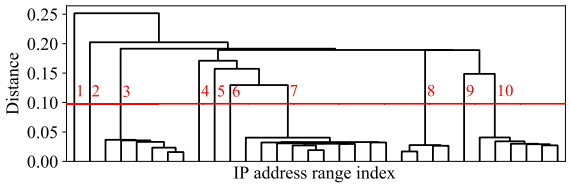
them "specially-configured" IP addresses, while the rest are "normally-configured" IP addresses.

Specially-configured IP addresses are difficult to recognize from the scanner's perspective. Scanning performance is expected to deteriorate as the proportion of specially-configured IP addresses increases. To quantitatively measure the impact of IP address relocation dynamics on scanning strategies, we conducted experiments under different IP address relocation policies, i.e., different proportions of specially-configured IP addresses. In our experimental setting, we assume that each specially-configured IP address belongs to a newly initialized IP pool, and these addresses are independent.

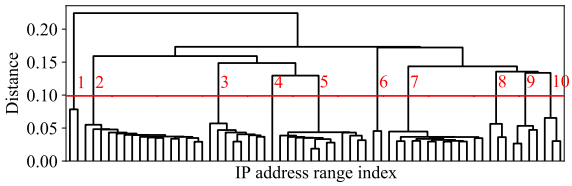
Fig. 8(a) and Fig. 8(b) depict the performance results of employing different proportions of specially-configured IP addresses in Scenario A and Scenario B, respectively. In Scenario A, no trending impact of the proportion of specially-configured IP addresses on the average RP score can be observed for each strategy. This is probably due to the maximized randomness of IP-device mapping dynamics. In contrast, in Scenario B, a discernible degradation in performance can be observed. That is, with every 1% increase in the proportion of specially-configured IP addresses, the average RP score decreased by approximately 0.6%. This experimental evidence demonstrates that using specially-configured IP addresses leads to a decline in RP scores for the scanning strategies. However, this decrease is insignificant when the proportion of specially-configured IP addresses remains small. Additionally, the "God's view scanning" strategy possesses precise knowledge regarding the exact time when a device changes its IP address, hence achieving optimal performance across all scanning strategies.

4) IP Pool Estimation: We use hierarchical clustering to demonstrate the effect of IP address pool estimation. The advantage of hierarchical clustering is that we can observe the whole clustering process and choose a more realistic number of clusters. We use our proposed scanning strategies to collect scanning records in the clustering experiments. We find that the clustering results are not sensitive to the distance function, and hence we use the single-linkage clustering [37].

Fig. 9(a) shows the clustering result using the parameter settings in Table II. We initialize 8,192 IP addresses from 10 IP pools with 20 different types of devices. The smallest unit of clustering is an IP address range consisting of 256 IP addresses, i.e., a class C network, and the number of IP address ranges to cluster is 32. If correctly clustered, ten different clusters will be obtained. Fig. 9(a) is plotted after 100 rounds of scanning. The distance threshold interval for correct clustering is [0.041,0.129] in Fig. 9(a), and [0.079,0.129] in Fig. 9(b). If a threshold of 0.1 is selected, the IP address ranges would be clustered into ten pools, corresponding to the ten different IP pools that we initialized. After clustering, we derive a compact representation of the IP pool-device mutation intensity matrix. The size of the compact matrix is 69.7% smaller than that of the original matrix. Consequently, each element of the compact matrix has on average 3.2 times the number of IP addresses of the original matrix, resulting in more samples for accurate intensity estimation.



(a) Setting 1: the number of IP address ranges=32, scan rate=0.5



(b) Setting 2: the number of IP address ranges=64, scan rate=0.25

Fig. 9. IP pool estimation using hierarchical clustering. When a threshold of 0.1 is selected, the IP address ranges (i.e., class C networks) are accurately clustered into 10 IP pools in both settings (following Table II unless specified).

{ "ip": "117.[redacted]", "ports": [{"port": 21, "proto": "tcp", "status": "open", "reason": "syn-ack", "ttl": 244}] },
{ "ip": "117.[redacted]", "ports": [{"port": 21, "proto": "tcp", "status": "open", "reason": "syn-ack", "ttl": 244}] },
{ "ip": "117.[redacted]", "ports": [{"port": 21, "proto": "tcp", "status": "open", "reason": "syn-ack", "ttl": 115}] },
{ "ip": "1.[redacted]", "ports": [{"port": 21, "proto": "tcp", "status": "open", "reason": "syn-ack", "ttl": 54}] },

(a) Port scan using MASSCAN

{"ip":"61.[redacted]","data":{"telnet":{"status":"success", "protocol":"telnet","result":{"banner":"Login:","will":[{"name":"Echo", "value":1},{"name":"Suppress Go Ahead","value":3},{"name":"Terminal Type","value":24},{"name":"Negotiate About Window Size","value":31}, {"name":"Linemode","value":34},{"name":"Status","value":5}],"do": [{"name":"Terminal Type","value":24},{"name":"Negotiate About Window Size","value":31}],"wont":[{"name":"Echo","value":1}]}, "timestamp":"2022-[redacted]"}}}
{"ip":"125.[redacted]","data":{"telnet":{"status":"connection-timeout", "protocol":"telnet","timestamp":"2022-[redacted]", "error":"dial tcp [redacted] timeout"}}}

(b) Application-layer scan using ZGrab2

Fig. 10. An example of the scanning results using MASSCAN and ZGrab2. The text in the green box represents the scan result of a tuple of an IP address and a port.

VIII. Real-World Evaluation

The simulation-based evaluation has verified the feasibility of our system, but the IP-device mutation dynamics in reality may be different from those in the simulation. To evaluate the performance of our system in consideration of real-world IP-device mutation dynamics, we perform real-world evaluation.

A. Experiment Settings

The scanning system is built based on two famous scanning tools: MASSCAN [38] and ZGrab2 [3]. We use MASSCAN to do the port scan and ZGrab2 to do the application-layer scan. If a port of a host is found open by MASSCAN, it will be further scanned for identifying applications using ZGrab2.

Figure 10 shows an example of the scanning results using MASSCAN and ZGrab2. The port scan records list all IP addresses and their open ports, as shown in Figure 10(a). The application-layer scan records include time, IP address, protocol, status, banner, and protocol-specific fields, such as

"will" and "do" options of the Telnet protocol. Figure 10(b) shows the scan records using the Telnet protocol.

Note that our scanning experiment only uses port scanning and banner scanning of MASSCAN and ZGrab2, and does not include any form of vulnerability scanning or exploitation. Our experiment setting is legal in China. Cyber search engines similar to our setting, such as FOFA, HUNTER, and ZoomeEye, have been built in China in recent years. In other countries, cyber search engines, such as Shodan [1] and Censys [2], have also been developed. Although IP scanning appears legal in many countries, several ISPs prohibit scans, like US-based cable company Comcast [39]. In our experiment, only one scanner is used to scan 3,261,184 IP addresses with a very limited scanning rate (e.g., 300 packets per second). In other words, each IP address received less than ten packets per day on average. (This setting can scan up to around 40 times as many IP-device mapping mutations as random/sequential scanning while resulting in negligible overhead for IoT devices.)

People may have privacy concerns about network scanning. For example, one may be concerned that if an attacker scans a device frequently, the attacker may be able to infer the online/offline dynamics of the device and in turn the living habits of the device owner by observing IP-device mutations. However, our method will not cause privacy exposure for three reasons. First, the IP-device mutation does not necessarily mean that the device goes online/offline, and it may also be caused by the device changing its IP address. Therefore, attackers cannot determine whether a home security service is temporarily offline or has shifted to a new IP address, making physically infiltrating homes lack underlying support. Second, exposing fine-grained privacy, such as live habits, requires a high scanning rate, but our scanning rate is low. Last but not least, our method does not involve any mining of user privacy and will not increase existing cyber search engines' risk of privacy exposure. Also, we declare that the historical scanning data will not be saved in our experiment, nor be used to explore user privacy.

The scanning system scanned 3,261,184 IP addresses in a major city in China in March 2022. The ports include 21, 22, and 23; the application-layer protocols are Telnet, SSH, and FTP. To uniquely identify the device type that hosts the application (e.g., Telnet), we calculate the hash function of the fingerprint of the scanning results (e.g., banner, options, versions) to represent the device type behind the port (e.g., 21).

B. Performance Using Bucket-Based Online Learning Strategy

Because of its high scalability, the bucket-based online learning strategy is used in real-world scanning experiments, in comparison to random scanning and sequential scanning strategies. In the bucket-based online learning strategy, we leverage IP-port, IPs-port, IPs-port-device granularities to partition the task set into 3261184×3, $3261184 \times 3/256$, $3261184 \times 3 \times d/256$ tasks, respectively, and accordingly instantiates three bucket-based strategies (bucket-based-I/ bucket-based-II/bucket-based-III scanning). Here, 3,261,184 and 3 are the numbers of IPs and ports. 256 is the number of IPs in a class C network, and d is the number of device types (we define its maximum as 256).

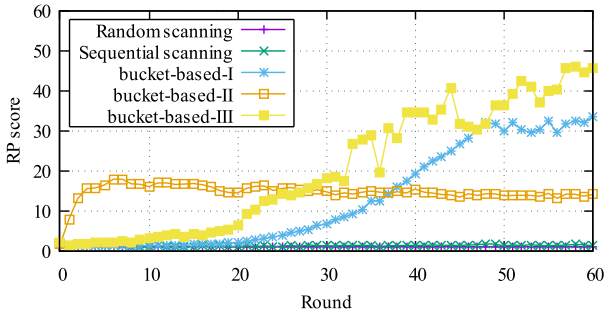


Fig. 11. Real world scanning performance using different strategies.

Fig. 11 shows the RP score of different strategies over 60 scanning rounds. The RP score of sequential scanning is between 1.1 and 1.4, whereas our strategy can achieve an RP score of around 45. This demonstrates that our system can scan up to around 40 times as many IP-device mapping mutations as random/sequential scanning. Also, at the beginning of the scan, the RP score of bucket-based-II scanning shows a significant rising trend and converges rapidly within five rounds. When the number of scanning rounds reaches 30, bucket-based-III scanning gradually outperforms bucket-based-II scanning. When the number of scanning rounds reaches around 40, bucket-based-I scanning also surpasses bucket-based-II scanning. When the number of scanning rounds reaches around 50, the growth rates of RP scores of all strategies slow down.

Among the three bucket-based strategies instantiated based on task granularity, bucket-based-I scanning converges the slowest, since it has the finest task granularity. However, at the cost of slow convergence, it can finally achieve a higher RP score than bucket-based-II scanning. When we look closer at bucket-based-II scanning and bucket-based-III scanning, the latter constantly outperforms the former due to its more fine-grained partition of tasks.

The real-world scanning results in several times larger RP scores than the simulation-based scanning. This reason is that in real-world scanning IP-device mutation probabilities across different tasks are diversely distributed, while in simulation-based scanning the probabilities are uniformly distributed. Our strategy is more beneficial when IP-device mutation probabilities across different tasks are more diversely distributed. Fig 12 shows the number of tasks in each bucket under the bucket-based-I strategy after 60 rounds of scanning. We observe that most tasks are concentrated around the 125th scan rate level, which slightly decreases from the initial 128th scan rate level due to the less frequent mutation. In addition, there is a small peak at the 255th scan rate level, indicating tasks in this bucket have quick IP-device mutations.

IX. Discussion

The key to smartly scanning IoT devices is to keep the system continuously learning historical scanning results. Our work provides a systematic solution. Despite its promising performance, several problems need to be discussed to make the solution appropriate and better used.

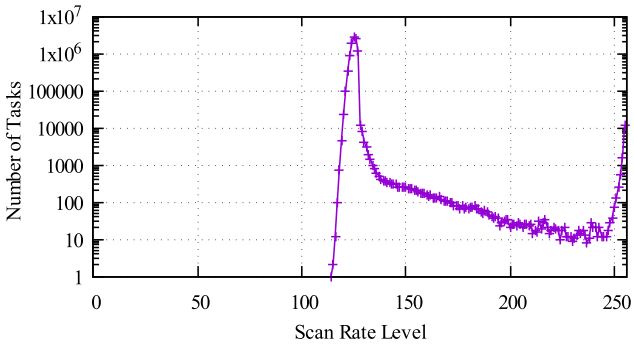


Fig. 12. The number of tasks in each bucket (i.e., scan rate level) under the bucket-based-I scanning strategy after 60 rounds of scanning.

First, different strategies may be used in combination with each other to achieve better performance than any single strategy. For instance, in Fig. 11, we can use the bucket-based-II strategy first and then switch to the bucket-based-III strategy, to accomplish both fast convergence and high RP scores. The flexible and seamless transition between strategies can be realized via parameter switching, that is, using the parameters of one strategy to initialize the parameters of another.

Second, besides using the number of mutations as a performance metric to optimize, other metrics like the average scanning record valid time could also be used. The scanning record valid time is the actual lifetime that the record is true in reality. Although the average scanning record valid time is significantly related to the number of mutations, it is more focused on the scanning timeliness. However, obtaining the record valid time is extremely challenging given the limited scanning rate for a certain task because it relies on the detailed timing information of IP-device mapping mutations. If IP-device mutations follow a homogeneous Poisson process, we can derive the optimal scanning rate to maximize the average scanning record valid time, as shown in Appendix. When IP-device mutations follow a non-homogeneous Poisson process, the integral form of the average record valid time hinders analytically deriving the optimal scanning rate, which we leave for future study.

Last, our bucket-based online learning strategy boosts system scalability by discretizing scan rates that are originally continuous variables into discrete scan rate levels. This strategy actually considers the scan rate of a task in a certain period of time constant, with an underlying assumption that the arrival rate of IP-device mutations is constant during that period. Intuitively, this assumption is arguably reasonable because in practice the network environment of most devices is stable in a short period of time.

Cyber search engines are built upon IP scanning. The legal ramifications of IP scanning have been controversial. This issue also draws many debates. However, many laws, such as United States federal laws, do not explicitly criminalize IP scanning [40]. In addition, IP scanning without actual intrusion is legal in many countries. Sometimes, unauthorized IP scanning is against the provider's acceptable use policy. If a scan is noticed, a more frequent occurrence is that the target network will send a complaint to the network service provider initiating the scan.

Note that our study is not an improvement of the ARP protocol. The ARP protocol works on the local area network, while the cyber search engines perform scanning by sending probes (e.g., IoT-specific HTTP requests) on the wide area network to determine IoT device types according to the probes' responses. Our approach focuses on the scan scheduling algorithms to help cyber search engines in the IPv4 space.

We use class B and class C IP space in the paper. Most local networks use CIDR today. However, it is tough for a scanner to know the configuration of all IP pools worldwide. Therefore, we assume that a segment of consecutive IP addresses (e.g., class C IP space) shares a similar intensity matrix since the class C network generally does not further divide into smaller IP pools.

In the future, there is no doubt that IPv6 will replace IPv4. However, at present, many devices on the Internet still use IPv4. According to Wilhelm's research, IPv6 adoption at the end user side is between 30% and 40% globally, with significant differences between regions and countries [41].

X. Conclusion

Scanning IoT devices in a principled way is an important problem. We made the first step toward investigating this problem based on a real-world global IoT scanning platform. Our large-scale measurement study revealed that both the device type and IP address pools are related to the IP-device mapping dynamics. Inspired by this observation, we designed a system capable of smartly scheduling scans for IoT devices. The proposed system can achieve a reinforcement learning-based continuous scanning decision making process using both online learning and batch learning strategies.

Through extensive experiments, we demonstrated that our system could generally capture significantly more IP-device mapping mutations than random and sequential scanning, and approach the God's view strategy. We revealed the two key parameters affecting the performance of different strategies, i.e., the scan rate and the proportion of devices to IP addresses. We found that, as the number of IoT devices grows, our system would become far more advantageous than random and sequential scanning. The real-world experiments show that our system can scan up to around 40 times as many IP-device mapping mutations as random/sequential scanning.

Appendix

A. Record Valid Time Under Homogeneous Poisson Process

We assume that for one task, the generation of mutations is completely consistent with a homogeneous Poisson process. Suppose that the first mutation time is S_0. Then, probability density of S_0 is

$$f_{S_0}(t) = \lambda e^{-\lambda t}, \tag{10}$$

where t represents time.

Suppose we execute one task at a fixed frequency f. Then, the same process is repeated at period T, where $T = 1/f$. And we only need to focus on the changes in one cycle. In one cycle, the valid time D of one record can be expressed as

$$D = \begin{cases} S_0 \text{ if } S_0 \leq T, \\ T \text{ if } S_0 > T. \end{cases} \tag{11}$$

The cumulative distribution function of D is

$$F(t) = \begin{cases} 0 & t < 0, \\ 1 - e^{-\lambda t} & \text{if } 0 \le t < T, \\ 1 & T \le t. \end{cases} \tag{12}$$

In a cycle, the proportion of the average scanning record valid time is

$$\frac{E(D)}{T} = \frac{1}{T}\left[Te^{-\lambda T} + \int_0^T t\lambda e^{-\lambda t} dt\right]. \tag{13}$$

The simplified formula is

$$\frac{E(D)}{T} = \frac{1}{\lambda T} - \frac{1}{\lambda T} e^{-\lambda T} \tag{14}$$

For the entire scanning system, it aims to maximize the valid duration expectation summation of each task subset, and the sum of the scan rates is a constant. The formula of this optimization problem is

$$\max \quad \sum_{i \in \text{Tasks}} f_i E(D_i) \tag{15}$$

$$\text{s.t.} \quad S = \sum_{i \in \text{Tasks}} f_i, \tag{16}$$

$$f_i \ge 0, \qquad \forall i \in \text{Tasks}$$

in which i refers to different tasks. f_i is the scan rate of the ith task, and S represents the total scanning resources.

It can be proved that if the objective function (15) is to reach its maximum value, for any two different tasks, the derivatives of scan rates must be equal, that is

$$\frac{df_1 E(D_1)}{df_1} = \frac{df_2 E(D_2)}{df_2}. \tag{17}$$

The simplified formula is

$$\frac{1}{\lambda_1} + \left(\frac{1}{f_1} + \frac{1}{\lambda_1 f_1^2}\right) e^{-\frac{\lambda_1}{f_1}} = \frac{1}{\lambda_2} + \left(\frac{1}{f_2} + \frac{1}{\lambda_2 f_2^2}\right) e^{-\frac{\lambda_2}{f_2}} \tag{18}$$

Formula (18) is difficult to find an analytical solution. Fortunately, one can still calculate the numerical solution and apply it to the real-world scanning.

REFERENCES

[1] Shodan. (2021). *The Search Engine for the Internet of Things*. [Online]. Available: https://www.shodan.io/

[2] Censys. (2021). *Censys*. [Online]. Available: https://censys.io/

[3] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proc. ACM CCS*, 2015, pp. 542–553.

[4] ZoomEye. (2021). *ZoomEye Cyberspace Search Engine*. [Online]. Available: https://www.zoomeye.org/

[5] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *Int. J. Crit. Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, Jun. 2014.

[6] H. Al-Alami, A. Hadi, and H. Al-Bahadili, "Vulnerability scanning of IoT devices in Jordan using Shodan," in *Proc. 2nd Int. Conf. Appl. Inf. Technol. Developing Renew. Energy Processes Syst. (IT-DREPS)*, Dec. 2017, pp. 1–6.

[7] S. Torabi, E. Bou-Harb, C. Assi, E. B. Karbab, A. Boukhtouta, and M. Debbabi, "Inferring and investigating IoT-generated scanning campaigns targeting a large network telescope," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 1, pp. 402–418, Jan. 2022.

[8] X. Ma, J. Qu, J. Li, J. C. S. Lui, Z. Li, and X. Guan, "Pinpointing hidden IoT devices via spatial–temporal traffic fingerprinting," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 894–903.

[9] X. Ma et al., "Inferring hidden IoT devices and user interactions via spatial–temporal traffic fingerprinting," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 394–408, Feb. 2022.

[10] J. Qu et al., "Landing reinforcement learning onto smart scanning of the Internet of Things," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 2088–2097.

[11] T. Dai and H. Shulman, "SMap: Internet-wide scanning for spoofing," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2021, pp. 1039–1050.

[12] M. Hastings, J. Fried, and N. Heninger, "Weak keys remain widespread in network devices," in *Proc. Internet Meas. Conf.*, Nov. 2016, pp. 49–63.

[13] Z. Durumeric, "Fast internet-wide scanning: A new security perspective," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Michigan, Ann Arbor, MI, USA, 2017.

[14] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *Proc. Conf. Internet Meas. Conf.*, Oct. 2013, pp. 291–304.

[15] B. Genge and C. Enăchescu, "ShoVAT: Shodan-based vulnerability assessment tool for internet-facing services," *Secur. Commun. Netw.*, vol. 9, no. 15, pp. 2696–2714, Oct. 2016.

[16] S. Lee, S.-H. Shin, and B.-H. Roh, "Abnormal behavior-based detection of Shodan and Censys-like scanning," in *Proc. 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2017, pp. 1048–1052.

[17] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *Proc. USENIX Secur.*, 2013, pp. 605–620.

[18] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Palo Alto, CA, USA: Insecure, 2009.

[19] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 2, pp. 93–108, Feb. 2005.

[20] Z. Shamsi, D. B. H. Cline, and D. Loguinov, "Faulds: A non-parametric iterative classifier for internet-wide OS fingerprinting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 971–982.

[21] S. Gordeychik, D. Kolegov, and A. Nikolaev, "SD-WAN internet census," 2018, *arXiv:1808.09027*.

[22] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering Internet-of-Things devices," in *Proc. USENIX Secur.*, 2018, pp. 327–341.

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

[24] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2177–2184.

[25] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of IoT devices," in *Proc. Workshop Attacks Solutions Hardw. Secur.*, Jan. 2018, pp. 41–50.

[26] O. Soyer, K.-Y. Park, N. H. Kim, and T.-s. Kim, "An approach to fast protocol information retrieval from IoT systems," in *Advanced Multimedia and Ubiquitous Engineering*. Cham, Switzerland: Springer, 2017, pp. 226–232.

[27] J. O'Hare, R. Macfarlane, and O. Lo, "Identifying vulnerabilities using internet-wide scanning data," in *Proc. IEEE 12th Int. Conf. Global Secur., Saf. Sustainability (ICGS3)*, Jan. 2019, pp. 1–10.

[28] R. Droms, "Dynamic host configuration protocol," 1997. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2131.html

[29] L. Mamakos, D. Simone, R. Wheeler, D. Carrel, J. Evarts, and K. Lidl, "A method for transmitting PPP over Ethernet (PPPoE)," Tech. Rep., 1999. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2516.html

[30] W. Simpson, *Point-to-Point Protocol (PPP)*, document RFC 1661, RFC Editor, 1994.

[31] G. McGregor, "The PPP internet protocol control protocol (IPCP)," Tech. Rep., 1992. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1332.html

[32] R. Padmanabhan, A. Dhamdhere, E. Aben, and N. Spring, "Reasons dynamic addresses change," in *Proc. IMC*, 2016, pp. 183–198.

[33] NSFOCUS. (2021). *Reports*. [Online]. Available: https://nsfocusglobal.com/company-overview/resources/#reports

[34] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.
[35] G. C. M. Moura, C. Gañán, Q. Lone, P. Poursaied, H. Asghari, and M. van Eeten, "How dynamic is the ISPs address space? Towards internet-wide DHCP churn estimation," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.
[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[37] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *WIREs Data Mining Knowl. Discovery*, vol. 2, no. 1, pp. 86–97, Jan. 2012.
[38] R. D. Graham. (2014). *Masscan: Mass IP Port Scanner*. [Online]. Available: https://github.com/robertdavidgraham/masscan
[39] R. Trapkickin, "Who is scanning the internet?" in *Proc. Seminars FI IITM*, 2015, pp. 1–8.
[40] (2023). *Nmap Legal Issues*. [Online]. Available: https://nmap.org/book/legal-issues.html
[41] W. Rene. (2020). *IPv6 10 Years Out: An Analysis in Users, Tables, and Traffic*. [Online]. Available: https://labs.ripe.net/author/wilhelm/ipv6-10-years-out-an-analysis-in-users-tables-and-traffic/

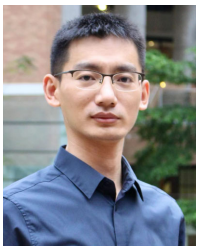
Jian Qu received the bachelor's degree in computer science and technology from the Special Class for the Gifted Young, Xi'an Jiaotong University, Xi'an, China, in 2019, where he is currently pursuing the Ph.D. degree with the MOE Key Laboratory for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering. His current research focuses on internet traffic analysis and cyber security.

Xiaobo Ma (Member, IEEE) received the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2014. He is a Professor with the MOE Key Laboratory for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University. He was a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University in 2015. He is a Tang Scholar. His research interests include internet measurement and cyber security.

Wenmao Liu received the Ph.D. degree in information security from the Harbin Institute of Technology in 2013. He is the Director of the Innovation Center and also a Leader of the Xingyun Laboratory, NSFOCUS Inc. He is the Co-Chair of Cloud Security Service WG, CSA. During the first two years in NSFOCUS, he was also with Tsinghua University, as a Post-Doctoral Researcher. His interests include cloud security, IoT security, data-driven analytics, and other new research areas of network security.

Hongqing Sang is the Manager of the Cyberspace Surveying and Mapping Laboratory, Innovation Center, NSFOCUS Inc. His interests include IoT security and cyberspace resource surveying and mapping.

Jianfeng Li received the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2018. He is an Assistant Professor with the MOE Key Laboratory for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University. He worked as a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University from September 2019 to June 2022. He has published a number of research papers in top conferences and journals, such as S&P, CCS, UNENIX security, NDSS, INFOCOM, TON, and TIFS. His research interests have centered on traffic analysis, privacy of mobile platform, network monitoring, AI security, and large-scale cyber security.



Lei Xue received the Ph.D. degree in computer science from The Hong Kong Polytechnic University. He is an Associated Professor with the School of Cyber Science and Technology, Sun Yat-sen University. His current research topics mainly focus on mobile and IoT system security, program analysis, and automotive security.

Xiapu Luo is a Professor with the Department of Computing, The Hong Kong Polytechnic University. His research focuses on mobile/IoT security and privacy, blockchain/smart contracts, network/web security and privacy, software engineering, and internet measurement. He has published papers in top security/software engineering/networking conferences and journals. His research led to eight best paper awards, including the ACM SIGSOFT Distinguished Paper Award in ICSE'21, the Best Paper Award in INFOCOM'18, the Best Research Paper Award in ISSRE'16, and several awards from industry.

Zhenhua Li (Senior Member, IEEE) received the B.S. and M.S. degrees from Nanjing University in 2005 and 2008, respectively, and the Ph.D. degree from Peking University in 2013, all in computer science and technology. He is an Associate Professor with the School of Software and BNRist, Tsinghua University. His research areas cover cloud computing/storage/download, big data analysis, content distribution, and mobile internet. He is a Senior Member of ACM.

Li Feng (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Xi'an University of Science and Technology, Xi'an, China, in 1997 and 2001, respectively, and the Ph.D. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, in 2005. He was a Visitor with Microsoft Research Asia (MSRA) joining the Microsoft Communication Protocols Project (MCPP), from June 2004 to August 2004. He is currently a Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. He has been a Post-Doctoral Research Fellow with the Department of Computer Science, Xi'an Jiaotong University, since September 2007. His current research interests focus on mobile ad hoc network and information security. He is a member of the IEEE Computer Society.

Xiaohong Guan (Life Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, in 1993. He is with the MOE Key Laboratory for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China. He is also a Cheung Kong Professor of systems engineering and the Dean of the Faculty of Electronic and Information Engineering. Since 1995, he has been with the Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Center for Intelligent and Networked Systems, Tsinghua University. He is an Academician of Chinese Academy of Sciences.